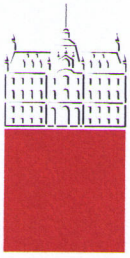
Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si



Št. naloge: 01655/2010
Datum: 05.04.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MARKO ILIĆ**

Naslov: **RAZŠIRITEV ANALITIČNE REŠITVE BI4DYNAMICS V MODULARNO OGRODJE POSLOVNE INTELIGENCE**

ENHANCEMENT OF BI4DYNAMICS ANALYTICAL SYSTEM INTO A BUSINESS INTELLIGENCE MODULAR FRAMEWORK

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V okviru diplomske naloge izdelajte načrt prenove arhitekture analitične rešitve za transakcijski sistem BI4Dynamics, ki bo temeljil na dinamičnem oziroma prilagodljivem podatkovnem modelu. Za potrebe načrta najprej analizirajte in opišite obstoječo arhitekturo ter izpostavite njene prednosti in slabosti.

Mentor:

prof. dr. Marko Bajec

Dekan:

prof. dr. Nikolaj Zimic

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Marko Ilić

Razširitev analitične rešitve BI4Dynamics v modularno ogrodje poslovne inteligence

DIPLOMSKO DELO

NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Marko Bajec

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01655/2010
Datum: 05.04.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MARKO ILIĆ**

Naslov: **RAZŠIRITEV ANALITIČNE REŠITVE BI4DYNAMICS V MODULARNO OGRODJE POSLOVNE INTELIGENCE**

ENHANCEMENT OF BI4DYNAMICS ANALYTICAL SYSTEM INTO A BUSINESS INTELLIGENCE MODULAR FRAMEWORK

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V okviru diplomske naloge izdelajte načrt prenove arhitekture analitične rešitve za transakcijski sistem BI4Dynamics, ki bo temeljil na dinamičnem oziroma prilagodljivem podatkovnem modelu. Za potrebe načrta najprej analizirajte in opišite obstoječo arhitekturo ter izpostavite njene prednosti in slabosti.

Mentor:

prof. dr. Marko Bajec

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani/-a **Marko Ilić**,
z vpisno številko **63040054**,

sem avtor/-ica diplomskega dela z naslovom:
Razširitev analitične rešitve BI4Dynamics v modularno ogrodje poslovne inteligence

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek) **prof. dr. Marko Bajec**

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne **13.10.2010** Podpis avtorja/-ice: _____

Zahvala

Za pomoč in nasvete pri izdelavi diplomske naloge se iskreno zahvaljujem mentorju prof. dr. Bajec Marku. Posebna zahvala gre sodelavcem v podjetju NPS d.o.o, ki so sodelovali pri nadgradnji rešitve BI4Dynamics. Za moralno podporo se zahvaljujem tudi družini.

Kazalo

Kazalo slik

Kazalo tabel

Seznam uporabljenih kratic in simbolov

1. .NET: Microsoftovo okolje za razvoj spletnih storitev in drugih programskih komponent [8]
2. AMO: objekti za delo z analitično bazo (angl. Analysis Management Objects) [9]
3. ETL: pridobivanje, preoblikovanje in nalaganje (angl. extract, transform, load)
4. MEF: modularno razširljivo ogrodje (angl. modular extensibility framework)
5. OLAP: sprotna analitična obdelava podatkov (angl. online analytical processing) [8]
6. SOAP: standard za spletne storitve, ki temelji na XML (angl. simple object access protocol) [8]
7. SQL: strukturiran povpraševalni jezik za delo s podatkovno bazo (angl. structured query languange) [8]
8. UDM: univerzalni dimenzionalni model (angl. universal dimensional model)
9. XML: razširljiv označevalni jezik (angl. extensible markup language) [8]
10. XMLA: razširljiv označevalni jezik za analitični strežnik (angl. extensible markup language for Analysis)

1 Povzetek

BI4Dynamics je analitična rešitev poslovne inteligence za transakcijski sistem Dynamics NAV, ki temelji na razvojni platformi SQL strežnika in ogrodju .NET. Analitični sistem sestavlja namestitveni čarovnik, ki v zaporedju korakov na SQL strežnik namesti podatkovno skladišče in analitično bazo. Podatki se s pomočjo procesov ETL prečrpajo v podatkovno skladišče na področje priprave podatkov, kjer se prečistijo in pripravijo za ustrezne analize, na koncu pa shranijo v prezentacijskempodročju podatkovnega skladišča. Analitična baza vsebuje nabor dimenzij in kock, ki črpajo podatke iz prezentacijskega področja podatkovnega skladišča. Arhitektura analitičnega sistema BI4Dynamics je bila zasnovana tako, da ni predvidela modularne strukture podatkovnega modela. Pri namestitvi rešitve BI4Dynamics se je vedno namestilo 6 standardnih analitičnih poslovnih področij, morebitna dodatna področja pa je bilo potrebno v sistem integrirati ročno. S širitvijo prodaje analitičnega sistema BI4Dynamcs na svetovni trg se je pojavila potreba po prilagoditvi podatkovnega modela, da bo na enostaven in hiter način omogočal namestitev in vpeljavo poljubnih modulov.

Pri analizi obstoječega modela sem najprej opisal celotno arhitekturo, kjer sem izpostavil ključne arhitekturne značilnosti, ki jih je pri prenovi potrebno zadržati in lociral določene strukturne elemente, ki jih nova arhitektura ne bo več potrebovala. Največ sprememb je doživelo področje priprave podatkov. Proces gradnje tega področja je bil v stari arhitekturi preveč kompleksen, da bi omogočal modularno gradnjo podatkovnega skladišča. V novi arhitekturi sem zato minimiziral nabor nastavitvenih tabel in procedur, ki so potrebne za izgradnjo področja priprave podatkov.Prezentacijsko področje podatkovnega skladišča je zaradi značilnosti dimenzijskega modela, ki že v osnovi zagotavlja modularnost, doživel malo sprememb. Objekti tega področja so doživeli le nekaj lepotnih popravkov v obliki novega načina poimenovanja posameznih objektov. Nova arhitektura je tako prijaznejša do razvijalcev analitičnih področij BI4Dynamics, saj jim na SQL strežniku omogoča enostavno in hitro iskanje med seboj povezanih objektov.

V stari različici sistema BI4Dynamics se je celotna analitična baza namestila s pomočjo predpripravljenega dokumenta XMLA, ki ga je namestitveni čarovnik posredoval analitičnemu strežniku. Nova arhitektura strukturira vse dimenzijske objekte in objekte kock v XML strukturirane objekte, ki se serializirajo v objekte AMO ogrodja .NET. Ti objekti se nato s pomočjo novega .NET klienta, ki skrbi za namestitev analitičnega sistema, namestijo na analitični strežnik.

Vsi objekti podatkovnega skladišča in analitične baze, ki tvorijo BI4Dynamics modul (dimenzija ali kocka) so v novi arhitekturi strukturirani v dokumente XML, ki zagotavljajo podporo večim različicam istega objekta. Za namestitev pravilne različice objekta in pravilni vrstni red nemestitve modulov skrbi ogrodje MEF, ki je implementirano v .NET klientu.

Prenovljen podatkovni model BI4Dynamics je prinesel koristi tako na prodajnem področju kot tudi na tehničnem področju. Prodaja se z novim paketnim modelom prodaje enostavno prilagaja potrebam svetovnega trga, medtem ko nova arhitektura omogoča enostavno širitev tudi na druge transakcijske vire, kot sta Dynamics CRM in Dynamics AX.

Ključne besede:

BI4Dynamics, podatkovno skladišče, analitična baza, podatkovni model, arhitektura, Microsoft Dynamics NAV

2 Abstract

BI4Dynamics is a business intelligence solution for the Microsoft Dynamics NAV transactional system based on the SQL Server platform and .NET Framework. Step-by-step analytical system's deployment wizard deploys a fully functional data warehouse and analytical database on the SQL Server. The NAV data is pumped into the staging area of the data warehouse by a group of ETL processes where it gets cleaned, restructured and in the end stored into the presentation area of the data warehouse. Analytical database consists of a group of dimensions and cubes, which reads the data from the data warehouse presentation area. The BI4Dynamics architecture was not designed in a way to provide a modular structured data model. Therefore, every BI4Dynamics implementation comes with six standard analytical areas, which are deployed by the deployment wizard. All additional analytical areas need to be integrated into the BI4Dynamics analytical system manually. When BI4Dynamics entered the worldwide market, the need of a modular data model, which allows deployment of any group of BI4Dynamics modules using the deployment wizard, grew stronger and stronger.

In the analysis of the old BI4Dynamics data model, I described the entire architecture where I pointed out the key architectural qualities that need to be kept in the new architecture. I also pointed out some obsolete architectural parts that need to be discarded in the new architecture. The biggest changes were implemented in the data warehouse staging area architecture. The process of building the staging area in the old architecture was too complex to provide modular deployment of the data warehouse. In the new architecture, I minimized the number of setup tables and procedures which are used during the deployment of the staging area. The presentation area of the data warehouse did not undergo many architectural changes due to the nature of the dimensional data model, which provides modularity by default. The presentation area objects underwent some minor changes in the form of changing the naming convention, which is more user-friendly in terms of locating the objects on the SQL Server in a timely fashion.

In the old BI4Dynamics version, the entire analytical database was deployed from an XMLA document, which was sent to the analysis services server by BI4Dynamics deployment wizard. The new architecture manages all dimensions and cubes into XML structured objects, which can be serialized into .NET Framework AMO objects. These objects are then deployed on the analysis services server by the new BI4Dynamics .NET client.

All data warehouse and analytical database objects of the new BI4Dynamics data model, which conflate into the BI4Dynamics module (dimension or cube), are structured into XML files, which provide multi-versioning ability for each object. Deployment of the correct version of the object and the right deployment order is managed by the MEF framework implemented in the BI4Dynamics .NET client.

The new BI4Dynamics data model brought benefits both inthe trade and technical fields. The sales teams use a new tailored-based package sales model that allows adapting to the needs of the worldwide market. From the technical point of view, the new architecture provides a

platform which can be easily expanded to other transactional data sources, such as Dynamics CRM and Dynamics AX.

Keywords:

BI4Dynamics, data warehouse, analytical database, data model, architecture, Microsoft DynamicsNAV

3 Uvod

V podjetju NPS d.o.o. smo razvili poslovno rešitev BI4Dynamics, ki nad podatki Microsoft Dynamics NAV zgradi celovit analitični sistem, ki temelji na podatkovnem skladišču in množico OLAP kock. Pri tem smo se omejili izključno na razvojno platformo, ki jo ponuja Microsoft, tako da aplikacijski del rešitve temelji na ogrodju .NET, podatkovni del pa na SQL strežniku.

Analitični sistem BI4Dynamics je bil prvotno razvit za potrebe slovenskega trga. Načrtovan in zasnovan je tako, da se s pomočjo namestitvenega čarovnika postavi podatkovno skladišče in analitična baza za 6 poslovnih področij transakcijskega sistema Dynamics NAV. Kasneje so potrebe trga narekovale širitev sistema na 4 dodatnanestandardna področja. Prvotna arhitektura sistema BI4Dynamics širitve na dodatna področja ni predvidela, zato so se vsa dodatna področja nameščala ročno brez uporabe namestitvenega čarovnika. S selitvijo na svetovni trg se je to izkazalo za veliko arhitekturno pomanjkljivost, saj je arhitektura poleg časovno zamudnega nameščanja dodatnih modulov, omejevala uporabo različnih prodajnih modelov in posledično slabo prilagajanje določenim trgom. V podjetju smo se zato odločili za nadgradnjo celotne arhitekture BI4Dynamics, ki bo omogočala namestitev poljubnega števila modulov s pomočjo namestitvenega čarovnika.

V procesu razvoja nove arhitekture BI4Dynamics sem bil zadolžen za nadgradnjo podatkovnega modela podatkovnega skladišča in analitične baze. Nadgradnje sem se lotil tako, da sem najprej analiziral obstoječo arhitekturo in izpostavil pomanjkljivosti, ki so se morale z novo arhitekturo odpraviti. Iz analize sem izključil nekatere dele podatkovnega modela, ki se nanašajo na funkcionalnosti, ki smo jih z novo različico ukinili. V diplomskem delu sem celotno analizo skušal predstaviti skozi opis stare arhitekture.

Pri načrtovanju nove arhitekture sem poskušal ohraniti vse ključne elemente stare arhitekture in hkrati implementirati vse potrebne popravke za zagotavljanje popolne modularnosti podatkovnega modela. Pri tem sem največjo pozornost namenil področjem, ki sem jih izpostavil v analizi stare arhitekture. V drugem delu diplomskega dela sem predstavil vse konceptualne spremembe stare arhitekture.

4 Platforma BI4Dynamics

BI4Dynamics je analitičnisistem, ki na osnovi tehnologije Microsoft SQL Server, odpravlja analitične pomanjkljivosti ERP sistema Microsoft Dynamics NAV. Tako kot vsak transakcijski sistem je tudi NAV optimiziran za hitro in učinkovito zajemanje podatkov. To vrstna optimizacija je možna samo z zelo razdrobljenim podatkovnim modelom, ki s številnimi tabelami minimizira redundanco podatkov.Razdrobljenost podatkov, v nasprotju spisanjem, predstavlja zmogljivostni problem pri izvajanju poizvedb.[1] Pri podatkovnih analizah so namreč zelo številčne poizvedbe SQL, ki združujejo podatke iz velikega števila tabel. Pri teh poizvedbah se uporabljajo zmogljivostno najzahtevnejšestične operacije, ki so pri obsežnih tabelah zelo potratne z resursi.

Rešitev leži v prestrutkuriranju podatkov v podatkovno skladišče, ki temelji na dimenzijskem modelu. Tak model močno poenostavi strukturo relacijske podatkovne baze, kar omogoča hitrejše izvajanje podatkovnih poizvedb. S prestrukturiranjem podatkovnega modela v dimenzijski model se pojavi nekaj podatkovne redundance, vendar to ne predstavlja večjega problema, saj analitični sistemi niso namenjeni zajemanju podatkov.

Podatkovno skladišče BI4Dynamics je podatkovni vir za nabor t.i. OLAP kock. Vsaka kocka vsebuje analitične podatke za eno od šestih poslovnih področij Dynamics NAV. Za OLAP kocke je značilno, da vsebujejo predpripravljene hierarhično urejene opisne podatke in predpripravljene izračune agregatov transakcijskih podatkov po posameznih hierarhijah. Na ta način poslovni uporabniki z uporabno ustreznih analitičnih orodij, ki se povezujejo na OLAP kocke, pridejo do ustreznih podatkovnih analiz na zelo hiter in enostaven način.

Glavne komponente analitičnega sistema BI4Dynamics so:

1. Namestitveni čarovnik
2. Procesi ETL
3. Podatkovno skladišče
4. OLAP kocke
5. Administrativni modul

Namestitveni čarovnik je .NET aplikacija, ki poskrbi, da se v nekaj korakih na SQL strežnik postavi polno funkcionalno podatkovno skladišče, na analitični strežnik nabor OLAP kock in med integracijske storitve paket ETL, ki skrbi za osveževanje podatkov v celotnem sistemu. V korakih namestitvenega čarovnika se nastavijo informacije o:

1. Strežniku podatkovnega skladišča BI4Dynamics
2. Strežniku analitične baze BI4Dynamics
3. Strežniku podatkovnega vira Dynamics NAV
4. Različici podatkovnega vira Dynamics NAV
5. Podjetjih znotraj vira Dynamics NAV
6. Konsolidacijskih valutah med podjetji
7. Seznamu dimenzij vira Dynamics NAV

Procesi ETL poskrbijo, da se podatki iz podatkovnega vira Dynamics NAV prenesejo v področje priprave podatkov podatkovnega skladišča BI4Dynamics. Ti podatki se nato prečistijo in prestrukturirajo z naborom pogledov SQL in s pomočjo polnitvenih procedur zapišejo v dimenzijske tabele in tabele dejstev prezentacijskega področja podatkovnega skladišča. Sledi še zadnji korak, ki poskrbi, da se podatki iz prezentacijskega področja prenesejo v dimenzije in OLAP kocke analitične baze BI4Dynamics.

Administrativni modul je .NET aplikacija, ki omogoča upravljanje z analitičnim sistemom BI4Dynamics. Poleg ponastavljanja nastavitev namestitvenega čarovnika modul omogoča:

1. Ponovno namestitev podatkovnega skladišča BI4Dynamics
2. Ponovno namestitev analitične baze BI4Dynamics
3. Ponovno namestitev paketa ETL BI4Dynamics
4. Polnjenje podatkovnega skladišča BI4Dynamics
5. Polnjenje analitične baze BI4Dynamics
6. Vpeljavo poljubne tabele in polja transakcijskega vira Dynamics v prezentacijsko področje podatkovnega skladišča BI4Dynamics

Vse komponente analitičnega sistema BI4Dynamics zagotavljajo platformo za izdelavo poljubnih analitičnih modulov. Celoten sistem temelji na tehnologiji SQL strežnika, kar pomeni, da je rešitev razširljiva. Razvoj analitičnega modula poteka v treh korakih:

1. Vpeljava potrebnih tabel in stolpcev vira v prezentacijsko področje podatkovnega skladišča
2. Razvoj transformacijskih pogledov ETL, dimenzijskih tabel in tabel dejstev
3. Razvoj dimenzij in OLAP kock na analitični bazi

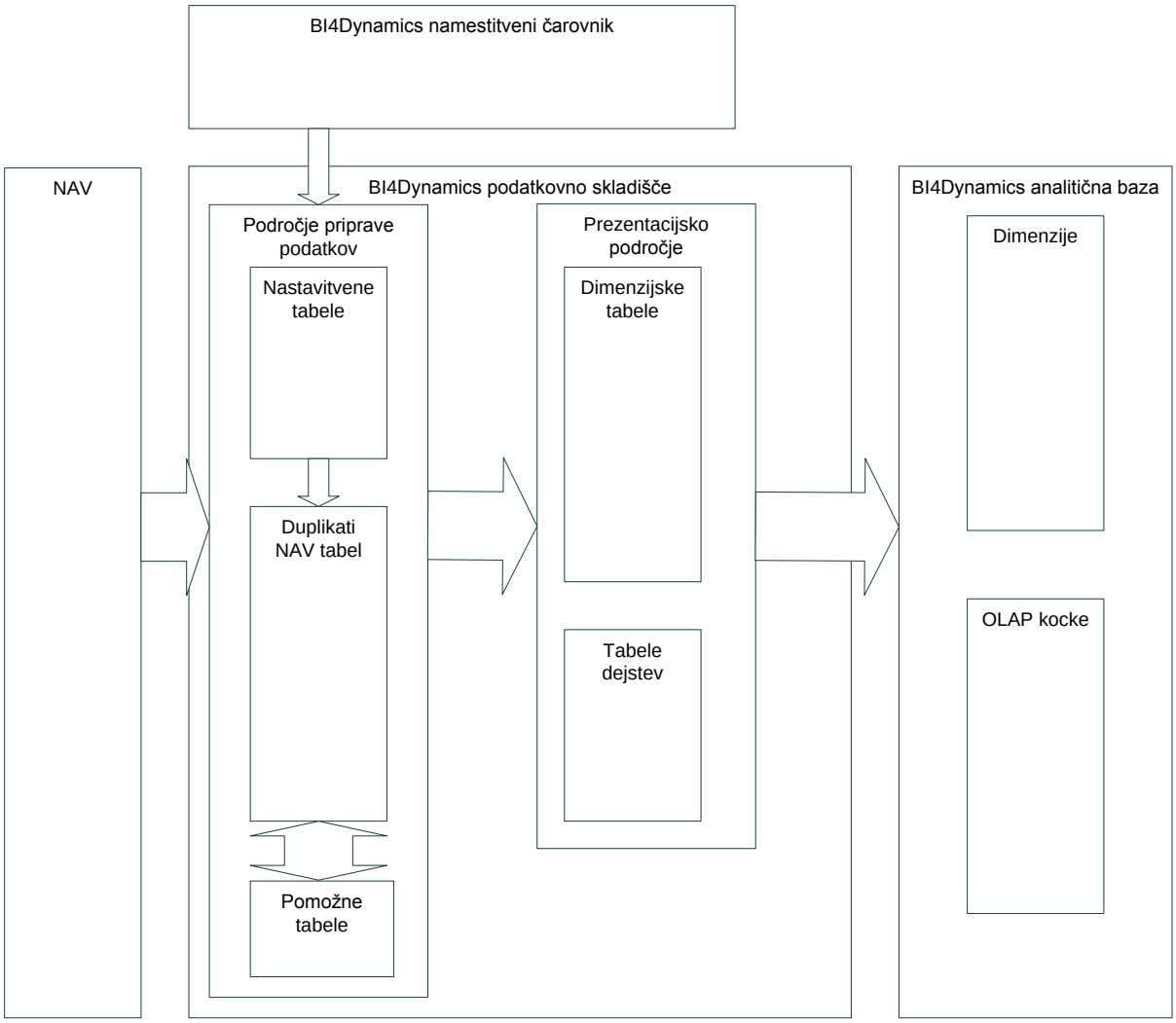
5 Arhitektura obstoječega podatkovnega modela

Osrednji in najpomembnejši del rešitve BI4Dynamics je njen podatkovni model. Zasnovan je na tehnologiji Microsoft SQL Server in sestavljata ga dve podatkovni bazi:

1. Transakcijska baza ali podatkovno skladišče
2. Analitična baza

Podatkovno skladišče je sestavljeno iz dveh področij:

1. Področja priprave podatkov
2. Prezentacijskega področja



Slika 1: Podatkovni model BI4Dynamics

5.1 Področje priprave podatkov

Področje priprave podatkov podatkovnega skladišča je prostor za shranjevanje podatkov in nabor t.i. procesov ETL.Podobno kot v kuhinji restavracije, kjer se surova neobdelana hrana obdela v okusne jedi, se v podatkovnem skladišču surovi transakcijski podatki obdelajo v obliko, ki je primerna za uporabniško poizvedovanje. Za podatke področja priprave podatkov je značilno, daposlovnim uporabnikom niso neposredno dostopni [2].

5.1.1 Nastavitvene tabele

Nastavitvene tabele so osnova za izgradnjo vseh ostalih tabel in procedur podatkovnega skladišča. Te tabele hranijo informacije o nastavitvah funkcionalnosti in virov iz katerih se v podatkovno skladišče črpajo podatki. Večino teh nastavitev vnese uporabnik med koraki namestitvenega čarovnika, nekaj pa se jih zabeleži v tabele samodejno.

Tabela 1: Nastavitvene tabele starega področja priprave podatkov

Tabela	**Opis**
setup.BasicSetting	Podatkovno skladišče BI4Dynamics lahko črpa podatke iz več NAV virov. Podprte so vse kombinacije NAV različic od različice 3.6 naprej. Pri tem pa je lahko transakcijska baza tipa Native oz. SQL. V nastavitveni tabeli so shranjeni podatki o različici, tipu (SQL ali Native) in imena NAV podatkovnih baz, iz katerih se podatki uvažajo v podtkovno skladišče. V primeru, da gre za Native bazo, se v tabelo shrani tudi pot do mape, kamor se izvažajo podatki v .txt datoteke. Iz teh datotek se podatki prenesejo v področje priprave podatkov. Za vsak SQL vir se v tabelo zabeleži tudi podatek o posebnem naboru znakov, ki so prisotni v imenih tabel in polj znotraj NAV klienta in se na SQL strežniku nadomestijo z znakom _. Pomanjkljivost te tabele leži v vmesniku za vnos posameznih virov. Vir se v tabelo zabeleži samo ob namestitvi z namestitvenim čarovnikom, ki pa ne podpira namestitve več virov. Tako je potrebno v primeru več NAV podatkovnih baz v nastavitveno tabelo vse ostale vire vnesti preko ukaza SQL. Z ukinitvijo podpore za NAV Native nova arhitektura ne bo več potrebovala določenih polj.
setup.ColumnList	Seznam stolpcev NAV tabel, iz katerih analitična področja BI4Dynamics črpajo podatke, so definirane v ločenem t.i. aktivacijskem XML dokumentu. Ta dokument vsebuje podatek, katere stolpce iz seznama vseh možnih stolpcev pripeljati v področje priprave podatkov. Vsi podatki iz tega dokumenta XML se shranijo v nastavitveno tabelo.
setup.Company	Vsak podatkovni vir NAV, iz katerega v podatkovno skladišče uvažamo podatke, lahko vsebuje podatke več podjetij. Rešitev konsolidacije podatkov večih podjetij v eno podatkovno bazo pa je pri NAV nekoliko specifična. Namesto da bi v tabelah obstajal poseben šifrant podjetij, NAV rešuje to problematiko s svojo množico tabel za vsako podjetje. Množica tabel je za vsa podjetja strukturno povsem enaka. Razlikuje se samo v imenih tabel na SQL strežniku, ki vključuje tudi ime podjetja. Ime tabele od imena podjetja ločuje znak $. Primer: dbo. CRONUS International Ltd_$Customer V eni podatkovni bazi NAV so lahko centralizirani podatki iz multinacionalnih podjetij, ki imajo svoja predstavništva v številnih državah. Za ta podjetja je značilno, da poslujejo v različnih valutah, kar pa onemogoča neposredno primerjavo podatkov. Vse zneske je zato potrebno, preden se zapišejo v tabele dejstev podatkovnega skladišča, preračunati v neko konsolidacijsko valuto.Ta lastnost pa močno vpliva na način, na katerega se podatki iz NAV prenašajo v področje za pripravo podatkov in kasneje v same dimenzijske tabele in tabele dejstev. V nastavitveno tabelo se zato shranijo podatki o imenih podjetij v

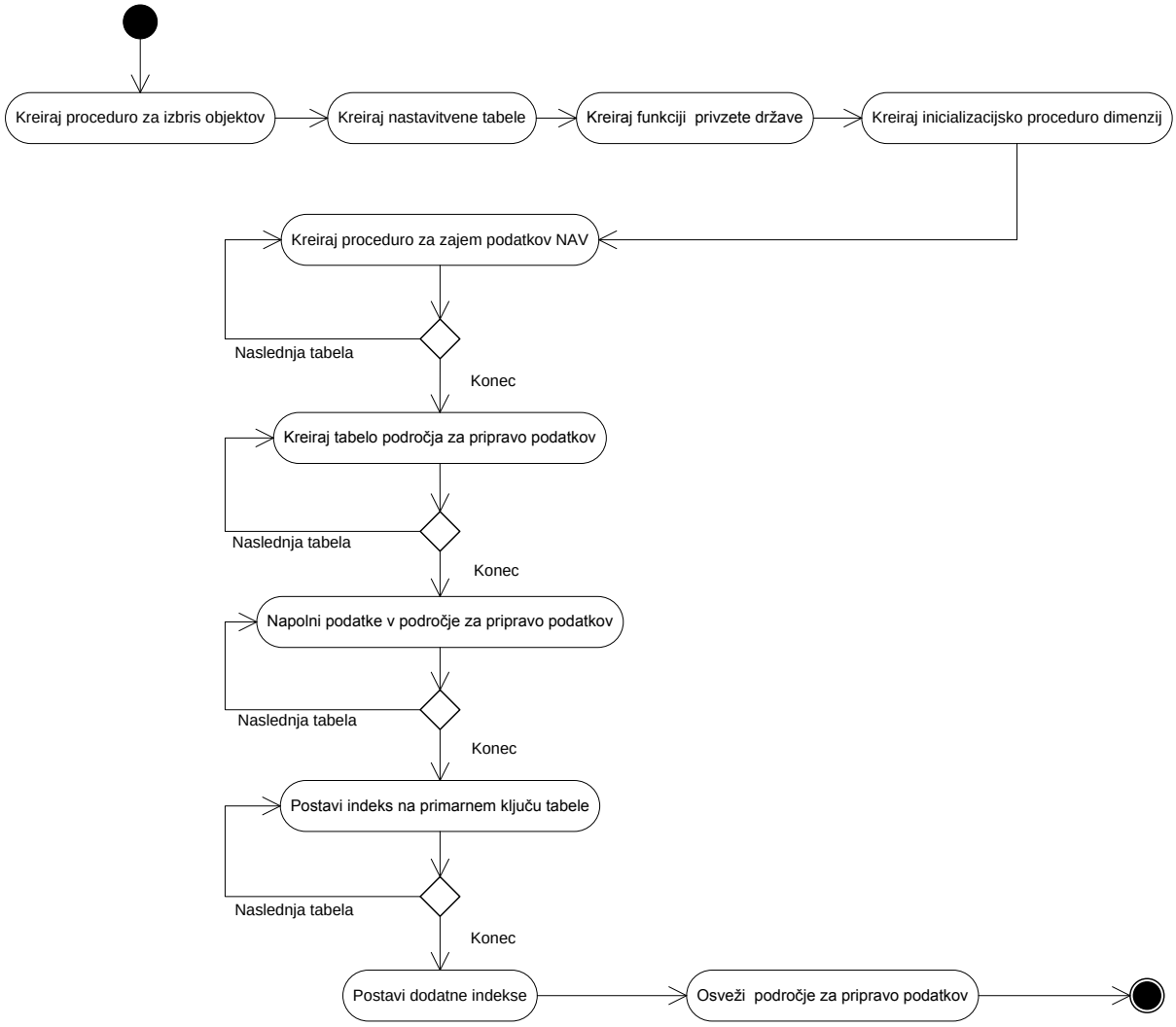
	posameznem viru in podatki o lokalni in dveh konsolidacijskih valutah, ki se uporabljata pri analizah. Ker se je v praksi izkazalo, da za potrebe analize podatkov zadostuje ena konsolidacijska valuta, v novi arhitekturi konsolidacija po dveh valutah ne bo več podprta.
setup.DefaultDimensionValue	Včasih se lahko zgodi, da v določeni dimenziji podatkovnega skladišča, za določeno transakcijo ne obstajajo opisni podatki. V teh primerih je potreben mehanizem, ki zagotavlja zmožnost analiziranja podatkov tudi za te transakcije. Poslovno nepravilno bi namreč bilo, da so te transakcije, zaradi pomanjkanja povezave v zvezdni shemi med tabelo dejstev in dimenzijsko tabelo, izključene iz analiz. Tak scenarij lahko povzroči odstopanja med zneski določenih postavk analitičnega sistema in zneski teh postavk v transakcijskem sistemu. Vsakršno odstopanje v podatkih med analitičnim sistemom in transakcijskim sistemom pa ponavadi najprej poruši zaupanje uporabnikov v analitični sistem. Za obravnavo teh primerov obstaja v rešitvi BI4Dynamics procedura, ki ob kreaciji dimenzijskih tabel poskrbi za vnos posebne vrstice v novo ustvarjeno dimenzijo. Ta vrstica ima v vseh dimenzijah šifrant 0 in za vsa polja dimenzije vsebuje privzeto vrednost, ki pa je odvisna od podatkovnega tipa posameznega polja. Privzete vrednosti za posamezne podatkovne tipe procedura prebere iz nastavitvene tabele. Z novo arhitekturo se je tabeli potrebno odpovedati in podatke o privzetih vrednostih prenesti v inicializacijsko proceduro.
setup.GlobalDimension	Transakcijski sistem NAV poslovnim uporabnikom omogoča knjiženje podatkov na prednastavljene dimenzije. Teh dimenzij pa ne gre zamenjevati z dimenzijami podatkovnega skladišča, saj v NAV dimenzije predstavljajo samošifrant, ki se poknjiži na glave dokumentov ali vrstice postavk. Vsaka NAV dimenzijaje v podatkovnem skladišču BI4Dynamics predstavljena s svojo dimenzijo. Za te dimenzije se je v obstoječi arhitekturi uporabljal izraz »globalna dimenzija«, ker so bile te dimenzije prisotne v vseh analitičnih področjih BI4Dynamics. Z novo arhitekturo se bo ta izraz ukinil. Za NAV dimenzije je značilno, da se lahko med podjetji vsebinsko povsem razlikujejo. Zato se za vsako podjetje vsakega podatovnega vira v tabelo shrani šifrant uporabljanih dimenzij.
setup.Log	Osveževanje podatkovnega skladišča BI4Dynamics poteka z zaporednim izvajanjem procedur SQL. Najprej se osveži področje za pripravo podatkov, nato pa se posodobijo dimenzijske tabele in tabele dejstev. Zadnji korak predstavlja osveževanje kock OLAP. Med izvajanjem posameznih procedur lahko včasih pride do napak, ki preprečijo polno osvežitev podatkovnega skladišča. Vzroki za te napake so ponavadi sistemske narave, včasih pa leži vzrok v pomankljivi integriteti podatkov samega vira. Za lažje odkrivanje in odpravljanje napak, se med izvajanjem osveževanja piše dnevnik, ki za vsako porceduro v tabelo zapiše podatek o

	začetku in koncu izvajanja ter status končnega uspeha. Pomanjkljivost obstoječe arhitekture pisanja dnevnika pa ne leži v sami vsebini tabele, temveč v samem mestu, kjer se tabela nahaja. Tabela je del podatkovnega skladišča, ki pa se lahko večkrat v celoti poruši in ponovno zgradi. Pri tem se lahko vsi podatki ponovno zapišejo v nastavitvene tabele. Izjema pa je tabela dnevnika, ki ob ponovni namestitvi podatkovnega skladišča izgubi podatke o predhodnem izvajanju osveževanja. Nova arhitektura mora zato predvideti in omogočiti ohranitev podatkov o izvajanju osveževanja tudi za scenarij ponovne namestitve celotnega podatkovnega skladišča.
setup.OriginalColumnList	Nabor stolpcev NAV tabel, ki jih je možno vključiti v podatkovno skladišče BI4Dynamics, je predefiniran v nastavitvenem XML dokumentu.Od tu se podatki zapišejo v nastavitveno tabelo. V teh podatkih so zajete informacije o imenih stolpcev in podatkovnih tipih stolpcev. Zelo pomembna so predvsem imena stolpcev, saj jemed NAV različicami možen pojav preimenovanja določenih vsebinsko enakih stolpcev. Poleg teh podatkov je v XML dokumentu prisotno tudi aktivacijsko polje, ki pove, ali je polje vključeno v podatkovno skladišče. Arhitekturno ima obstoječa rešitev vpeljave NAV stolpcev v področje priprave podatkov povsem enake omejitve kot nastavitvena tabela za vpeljavo tabel. Ti problemi pa so še nekoliko bolj izraziti, saj je število stolpcev, ki se jih vpeljuje v analitični sistem, veliko večje od števila tabel. Postopek je zato še bolj zamuden. Nova arhitektura bo, tako za tabele, kot tudi za stolpce zagotovila branje podatkov neposredno iz NAV vira. Na ta način z novo arhitekturo ta nastavitvena tabela ne bo več potrebna.
setup.OriginalTableList	Osrednji del področja priprave podatkov v podatkovnem skladišču BI4Dynamics predstavljajo duplikati NAV tabel. V praksi ne gre povsem za duplikate, saj tabele vsebujejo samo podmnožico polj iz NAV tabele. Poleg tega pa se v podatkovnem skladišču poskrbi za združevanje podatkov iz več virov in podjetij v eno tabelo področja priprave podatkov. Na ta način se izvrši prvo čiščenje podatkov, hkrati pa se minimizira obremenjenost vira in olajša nadaljnje čiščenje in transformiranje podatkov. Nabor NAV tabel, ki jih je možno vključiti v podatkovno skladišče BI4Dynamics, je predefiniran v nastavitvenem XML dokumentu, iz katerega se podatki zapišejo v nastavitveno tabelo. V teh podatkih so zajete informacije o primarnem ključu in imenu tabele v posameznih različicah NAV. Med NAV različicami je možen pojav preimenovanja določenih tabel. Poleg teh podatkov pa so v XML dokumentu prisotne tudi nastavitve polja za inkrementalno osveževanje tabele in podatek o tipu dimenzije NAV (dimenzija dokumenta, dimenzija postavke).

	Obstoječa rešitev vpeljave duplikatov NAV tabel v področje priprave podatkov je zelo omejena. NAV tabele, med katerimi lahko v administrativnem modulu uporabnik izbira, ko želi v podatkovno skladišče pripeljati novo tabelo, so predefinirane. Na ta način v podatkovno skladišče ni mogoče pripeljati nestandardnih NAV tabel, ki jih v NAV namestijo vertikalne rešitve in nestandardni moduli. Te tabele se sicer lahko pripelje v podatkovno skladišče, tako da se vse potrebne podatke izpolni ročno preko vmesnika administrativnega modula, vendar je ta postopek precej zamuden in neroden. Nova arhitektura mora odpraviti to pomanjkljvost in zagotoviti neposredno branje tabel iz NAV podatkovne baze. Na ta način z novo arhitekturo ta nastavitvena tabela ne bo več potrebna.
setup.Property	Rešitev BI4Dynamics je večinstančno okolje, kjer ima vsaka instanca svoje podatkovno skladišče s svojim naborom nastavitev. Večinstančno okolje se v praksi redko uporablja, vendar pa v primeru, da obstaja želja po ločenem razvojnem in produkcijskem okolju BI4Dynamics, se rešitev skriva v dveh instancah. Razvojna instanca je poligon za pripravo različnih dodelav in razvoj novih funkcionalnosti podatkovnega skladišča, ki se ob potrditvi sinhronizirajo tudi na produkcijsko instanco. Na ta način razvoj novih funkcionalnosti ne vpliva na odzivnost in dostopnost produkcijskega analitičnega sistema. V tabeli vsakega podatkovnega skladišča se nahajajo podatki o nastavitvah datumske dimenzije, inkrementalnega osveževanja in jezika. Za datumsko dimenzijo je potreben podatek o obdobju, določenim z začetnim datumom in obdobjem let v prihodnosti, v katerem se podatki analizirajo. Nastavitve inkrementalnega osveževanja so pomembe v primeru, ko se zaradi količine podatkov v NAV, podatkovno skladišče ne uspe pravočasno osvežiti znotraj časovnega okna, ki je za to predvideno. Za ta scenarij je predvidena nastavitev datuma, od katerega se podatki osvežujejo. Obstoječa rešitev inkrementalnega osveževanja se v praksi ni izkazala za uporabno, saj so se časi osveževanja minimalno skrajšali. Vzrok tiči v hitrosti brisanja podatkov zadnjega obdobja inkrementalnega osveževnja. Pri polnem osveževanju se podatki iz vseh tabel, z izjemo dimenzijskih, izbrišejo in nato ponovno napolnijo. V tem scenariju se uporablja operacija TRUNCATE, ki pa se izvede v trenutku. Pri inkrementalnem osveževanju se iz tabel podatki izbrišejo samo do določenega datuma in nato ponovno napolnijo. Pri tem scenariju se za brisanje uporablja operacija DELETE z dodatnim pogojem na datumu knjiženja. Ta operacija se na velikih tabelah izvaja zelo počasi, kar pa velikokrat postavi smiselnost uporabe inkrementalnega osveževanja pod vprašaj. V novi arhitekturi zato inkrementalno osveževanje ni več predvideno. Namesto tega se bo problem predolgega osveževanja reševal z optimizacijo skript

	podatkovnega skladišča in agregacijo starih podatkov.
setup.TableList	Obstoječa rešitev BI4Dynamics vsebuje 6 standardnih analitičnih področij, katera se s pomočjo namestitvenega čarovnika namestijo v podatkovno skladišče in analitično bazo. Seznam NAV tabel, iz katerih ta analitična področja črpajo podatke, so definirane v ločenem t.i. aktivacijskem XML dokumentu. Ta dokument vsebuje podatek, katere tabele iz seznama vseh možnih tabel pripeljati v področje priprave podatkov. Vsi podatki iz tega dokumenta XML se shranijo v nastavitveno tabelo.

V analitičnem sistemu BI4Dynamics področje priprave podatkov sestavljajo ekvivalenti tabel transakcijskega sistema NAV in nabor pogledov in procedur SQL, ki podatke iz teh tabel obdelajo in shranijo na prezentacijsko področje. Namestitev področja priprave podatkov se zgodi s pomočjo posebnih procedur SQL. Za vsako tabelo iz nastavitvenih tabel stolpcev in tabel, procedura izgradi ekvivalent tabele NAV in polnitveno proceduro, ki v to tabelo nalaga podatke z vira. Na tabele področja priprave podatkov se na koncu postavijo še indeksi, ki poskrbijo za hitrejše izvajanje procesa transformacije podatkov. Indeksirajo se stolpci primarnega ključa in stolpci, ki se pojavljajo v stičnih operacijah transformacijskih pogledov.



Slika 2: Namestitev področja za pripravo podatkov

Tabela 2: Koraki namestitve starega področja priprave podatkov

Aktivnost	**Opis**
Kreiraj proceduro za izbris objektov	Pri namestitvi področja priprave podatkov se najprej namesti procedura dbo.DropObject. S pomočjo te procedure se lahko izbriše vse objekte podatkovnega skladišča BI4Dynamics. Procedura skozi parameter dobi ime in tip objekta. Pri tem ime vključuje tudi ime sheme, tip objekta pa je lahko tabela, začasna tabela, pogled, procedura, funkcija, indeks ali shema. V kolikor v podatkovnem skladišču obstaja objekt s podanimi parametri, se objekt izbriše. Če objekt ne obstaja, procedura ne naredi ničesar. Procedura je ključnega pomena pri zagotavljanju zmožnosti neomejenega ponovnega nameščanja, tako posameznih modulov, kot tudi celotnega analitičnega sistema BI4Dynamics. Ta lastnost podatkovnega skladišča je zagotovljena s klicem procedure pred vsakim kreiranjem objekta. Parametra sta v tem primeru ime in tip

	objekta, ki ga želimo kreirati.
Kreiraj nastavitvene tabele	Nastavitvene tabele se namestijo kot del področja priprave podatkov. V tabele se prenesejo NAV nastavitve kot so podatkovni viri, podjetja in globalne dimenzije. Poleg NAV nastavitev pa so v tabelah tudi BI4Dynamics nastavitve. Te nastavitve vključujejo nastavitve tabel in stolpcev, podatkov o strežniku in podatkov o instanci. Pomanjkljivosti obstoječe arhitekture nastavitvenih tabel sta preveliko število tabel in neprimerno poimenovanje. Preveliko število tabel je posledica zastarelih postopkov gradnje področja priprave podatkov in ukinitve določenih funkcionalnosti kot sta podpora Native veriziji NAV in inkrementalnega osveževanja podatkov podatkovnega skladišča. V novi arhitekturi je zato potrebno ponovno premisliti vse koncepte gradnje področja priprave podatkov in tabele ustrezno skrčiti samo na potrebne podatke. Pri tem je potrebno tabele in polja poimenovati z enostavnimi imeni, ki jasno opišejo pomen tabele ali polja.
Kreiraj funkciji privzete države	Funkciji dwh.SetDefaultCountryCode in dwh.SetDefaultCountryName se v obstoječi arhitekturi pojavljata kot del področja priprave podatkov. Funkciji imata zelo pomembno vlogo pri zagotavljanju možnosti analiz s hierarhijo kupcev po državah. Za vsako podjetje znotraj NAV na karticah kupca in dobavitelja za domače kupce in dobavitelje pogosto ni podan šifrant države kupca ali dobavitelja. Funkciji za vse take kupce in dobavitelje prebereta podatek o državi iz nastavitvene tabele setup.Company in ta podatek pripeljeta na transformacijske poglede. Na ta način je odpravljen problem prikaza vseh teh kupcev in dobaviteljev pod vrstico N/A. Funkciji sta uporabni in služita svojemu namenu, vendar pa se arhitekturno nahajata na napačnem mestu. Funkciji odpravljata problem z dimenzijami kupca in dobavitelja, ki pa bosta v novi arhitekturi predstavljeni vsaka kot svoj modul. Ker se funkciji, z izjemo teh dveh modulov, ne uporabljata na nobenem drugem mestu v rešitvi BI4Dynamics, ju je v novi arhitekturi potrebno prestaviti med objekte teh dveh modulov.
Kreiraj inicializacijsko proceduro dimenzij	Procedura setup.InitDimension ob kreaciji dimenzijskih tabel poskrbi za vnos posebne vrstice v novo ustvarjeno dimenzijo. Ta vrstica ima v vseh dimenzijah šifrant 0 in za vsa polja dimenzije vsebuje privzeto vrednost, ki pa je odvisna od podatkovnega tipa posameznega polja. Procedura postopoma gradi ukaz INSERT, tako da se sprehodi čez vse stolpce dimenzijske tabele in pri tem primerja podatkovni tip trenutnega stolpca s tipom zapisanim v nastavitveni tabeli setup.DefaultDimensionValue. Od tam, kjer se podatkovna tipa ujemata, prebere privzeto vrednost in jo pripne v ukaz. Na koncu procedura ukaz tudi zažene in posledično je vpisana inicializacijska vrstica v dimenziji.
Kreiraj proceduro za zajem podatkov NAV	Procedure za zajem podatkov iz podatkovnih virov se zgradijo dinamično s pomočjo procedure setup.CreateFillStageProcedureForSqlVersion. Procedura za vsako

	NAV tabelo iz nastavitvenih tabel setup.TableList in setup.OriginalTableList postopoma gradi proceduro, ki iz tabele vseh virov, nastavljenih v tabeli setup.BasicSetting, črpa podatke v ekvivalent tabele v področju priprave podatkov. Osrednji del procedure je ukaz INSERT, v katerem se seznam stolpcev ciljne tabele in seznam stolpcev tabele vira sestavita povsem dinamično s pomočjo nastavitev v tabelah setup.TableList in setup.OriginalTableList. Način sestavljanja odseka FROM ukaza INSERT je odvisen od tipa tabele. V primeru, da gre za skupno tabelo podatkovne baze NAV, ki se ne multiplicira po podjetjih, se imenu tabele z leve strani prilepijo še ime strežnika, ime podatkovne baze in shema. V primeru, da je za tabelo v nastavitveni tabeli setup.OriginalTableList nastavljeno polje za inkrementalno osveževanje, se ukazu doda še pogoj WHERE, ki izloči podatke manjše od nastavljenega inkrementalnega datuma. Pri tabelah, ki se znotraj podatkovne baze NAV pomnožijo po vseh podjetjih, je gradnja odseka FROM ukaza INSERT nekoliko drugačna. Zanka, ki gradi proceduro, se iz obhoda po podatkovnih virih in tabelah spusti še en nivo nižje. Na tem nivoju za vsako podjetje znotraj ukaza INSERT naredi svoj UNION blok, v katerem bere podatke iz NAV tabele trenutnega podjetja. V odsek FROM se poleg podatkov o imenu strežnika, imenu podatkovne baze in imenu sheme pred ime tabele vrine še ime podjetja in znak $. V primeru, da je tudi za to tabelo v nastavitveni tabeli setup.OriginalTableList nastavljeno polje za inkrementalno osveževanje, se ukazu doda pogoj WHERE, ki izloči podatke manjše od nastavljenega inkrementalnega datuma. V novi arhitekturi je potrebno postopek kreiranja procedure za zajem podatkov iz NAV virov v področje priprave podatkov nekoliko poenostaviti. Zajem podatkov v področje priprave podatkov ne bo več potekalo preko ločene procedure za vsako tabelo, temveč preko skupne procedure, ki bo napolnila v enem obhodu vse tabele področja priprave podatkov. V primeru, da bo kot vhodni parameter procedure podan NAV šifrant tabele, se bo napolnila samo vhodna tabela. Na ta način se bo število objektov na podatkovnem skladišču občutno zmanjšalo in podatkovni model bo enostavnejši in preglednejši. Z novo arhitekturo je potrebno odpraviti tudi vse specifike povezane z inkrementalnim osveževanjem.
Kreiraj tabelo področja priprave podatkov	Podobno kot procedure za zajem podatkov iz virov, se tudi skripti za kreacijo tabel področja priprave podatkov, zgradijo dinamično s pomočjo temu namenske procedure. Procedura stage.CreateStageDatabase za vsako NAV tabelo iz nastavitvenih tabel setup.TableList in setup.OriginalTableList postopoma gradi skript SQL, ki v področju priprave podatkov zgradi ekvivalent NAV tabele. Osrednji del procedure je ukaz CREATE, v katerem se seznam stolpcev tabele sestavi povsem dinamično s pomočjo nastavitev v

	tabelah setup.TableList in setup.OriginalTableList. Na konec seznama se dodajo še polja CompanyName, C_id in BS_id, ki služijo kot šifrant podatkov vira in podjetja. Ta polja omogočajo združevanje podatkov iz več fizičnih tabel po podjetjih v eno tabelo področja priprave podatkov. V novi arhitekturi je potrebno postopek kreiranja procedure za kreacijo tabel področja priprave podatkov nekoliko poenostaviti. Prilagoditi je potrebno zajem tabel in stolpcev iz dveh nastavitvenih tabel na eno.
Napolni podatke v področje priprave podatkov	Ko se v področje priprave podatkov namestijo vse tabele namenjene hranjenju podatkov iz NAV transakcijskega sistema in procedure, ki v te tabele prenašajo podatke, je vse pripravljeno za prenos podatkov. Procedura stage.FillStage poskrbi, da se korakoma za vsako tabelo iz nastavitvenih tabel setup.TableList in setup.OriginalTableList zažene procedure, ki v področje priprave podatkov prenese podatke iz NAV tabele. V primeru, da gre za NAV tip Native, se podatki v področje priprave podatkov uvozijo iz .txt datotek, v katere NAV izvozi podatke. Z ukinitvijo podpori Native in opuščanjem ločenih polnitvenih procedur za vsako tabelo se proces prenosa podatkov v področje priprave podatkov močno poenostavi. Ta korak v novi arhitekturi zato ne bo več potreben.
Postavi indeks na primarni ključ tabele	V procesu transformiranja podatkov se podatki zajemajo iz več tabel področja priprave podatkov naenkrat. Pri tem se v transformacijskih pogledih pojavljajo številne stične operacije, ki pa so zmogljivostno najzahtevnejše operacije poizvedb SQL. Vpliv na zmogljivost je najbolj opazen pri stični operaciji med dvema zelo obsežnima tabelama. Večina tabel se povezuje s stično operacijo preko stolpcev primarnega ključa, nad katerim pa se, zaradi hitrejšega združevanja tabel, postavi indeks. Za postavitev indeksov na vse primarne ključe tabel področja priprave podatkov je poskrbljeno s proceduro stage.CreatePrimaryIndex.Procedura za vsako NAV tabelo iz nastavitvenih tabel setup.TableList in setup.OriginalTableList postopoma gradi skript SQL, ki na tabelo področja priprave podatkov postavi indeks. Osrednji del procedure je ukaz CREATE, v katerem se seznam indeksiranih stolpcev primarnega ključa sestavi povsem dinamično s pomočjo podatka o primarnem ključu, zapisanem v nastavitveni tabeli setup.OriginalTableList. Na konec seznama se dodata še polji C_id in BS_id, ki služita kot šifrant podatkov vira in podjetja.
Postavi dodatne indekse	Nekatere tabele se v transformacijskih pogledih med seboj združujejo tudi s pomočjo stolpcev, ki niso del primarnega ključa. Za optimizacijo izvajanja teh pogledov je potrebno indeksirati tudi take stolpce. V obstoječi rešitvi BI4Dynamics se ti indeksi postavijo s pomočjo procedure stage.CreateSpecialIndex. Osrednji del procedure je množica ukazov CREATE, kjer se na tabelah področja priprave podatkov postavijo dodatni indeksi. Ti indeksi so lahko tipa CLUSTERED ali tipa NONCLUSTERED.

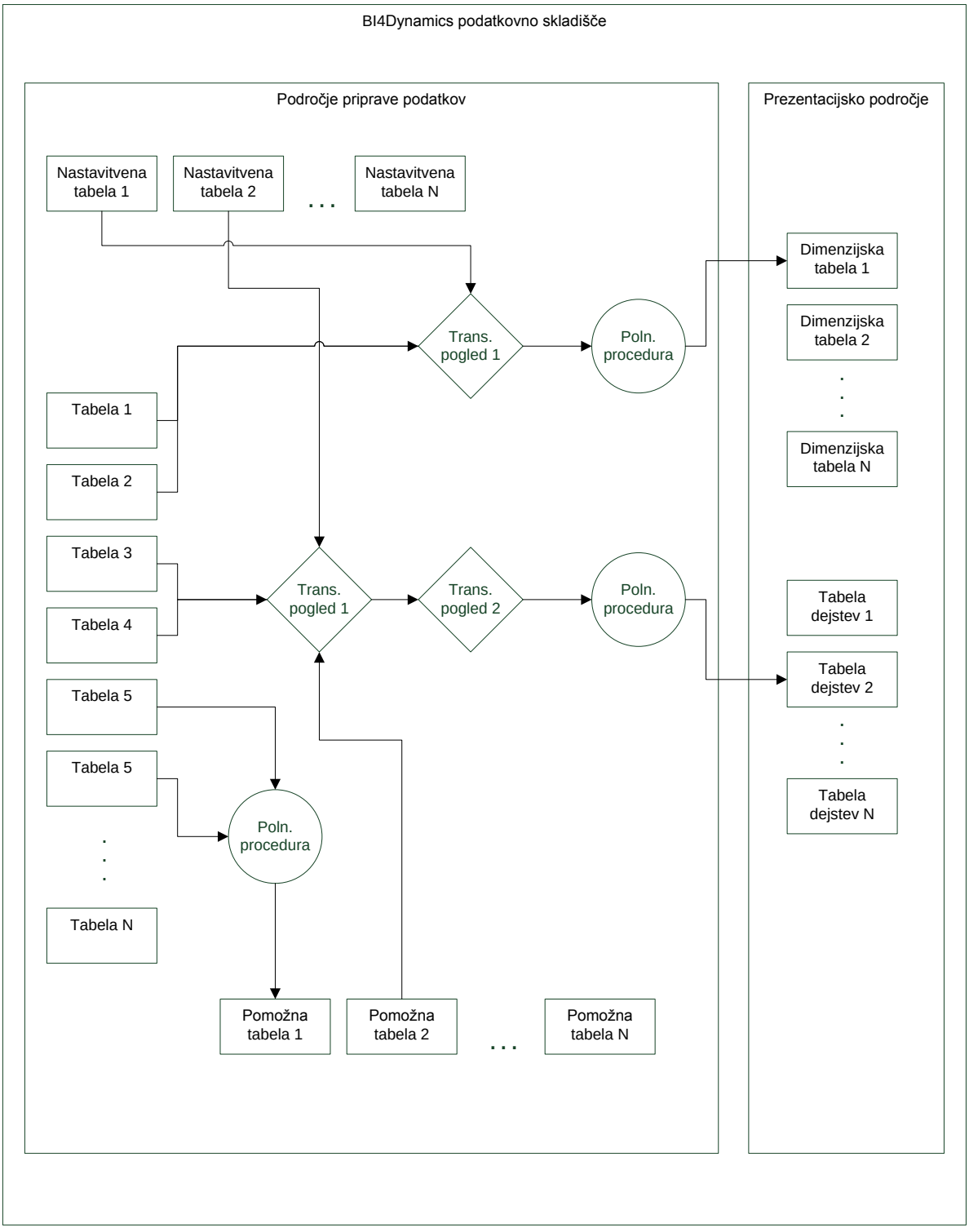
	Razlika je v tem, da je CLUSTERED indeks vedno urejen in strukturiran v drevo. Slabost obstoječe rešitve postavitve dodatnih indeksov je v tem, da procedura deluje povsem statično, saj je poslovna logika indeksiranih tabel in stolpcev definirana znotraj procedure. Nova arhitektura mora zagotoviti dinamično gradnjo dodatnih indeksov glede na vhodne parametre polj, zapisane v nastavitveni tabeli.
Osveži področje priprave podatkov	Procedura stage.UpdateStage je namenjena izgradnji in polnjenju vseh tabel področja priprave podatkov. Med njenim izvajanjem se najprej zaporedno zgradijo in napolnijo nastavitvene tabele. Za tem pa se zgradijo, napolnijo in indeksirajo še duplikati NAV tabel. V obstoječi arhitekturi je obstoj procedure smiselen zaradi popolne kontrole osveževanja podatkov preko SQL strežnika. Gledano s tehničnega vidika to pravilo ni potrebno, vendar pa je zaradi minimiziranja stroškov zunanjega .NET razvijalca, celotno podatkovno skladišče strukturirano tako, da se vsi popravki lahko implementirajo kar na SQL strežniku. Z razširitvijo internega razvojnega tima tudi na .NET razvijalca, to pravilo ni več obvezno. Zato se bo nova arhitektura odpovedala tej proceduri in izvajanje osvežitve področja priprave podatkov prenesla na .NET klienta.

5.1.2 Proces ETL

Prvi korak prenosa podatkov iz transakcijskega sistema v podatkovno skladišče je proces zajema podatkov. Pri tem se v področje priprave podatkov prenesejo samo podatki, ki so potrebni za podatkovno skladišče in nadaljne obdelave. Pri zajemu podatkov je zato ključnega pomena dobro poznavanje strukture podatkovne baze transakcijskega sistema [2].

Ko je proces zajema podatkov v podatkovno skladišče zaključen, se nad podatki začne proces transformacije. Proces obsega postopke čiščenja podatkov (popravki slovničnih napak, reševanje domenskih konfliktov, obravnavanje manjkajočih elementov in združevanje v standardne formate), združevanja podatkov iz večih podatkovnih virov, odpravljanja podvajanj v podatkih in določanje šifrantov podatkov znotraj podatkovnega skladišča. Vse transformacije podatkov se morajo zaključiti pred zadnjim procesom področja priprave podatkov – procesom shranjevanja podatkov [2].

Shranjevanje podatkov iz področja priprave podatkov podatkovnega skladišča zajema nalaganje podatkov potrjene kvalitete v nabor dimenzijskih tabel in tabel dejstev. Po uspešnem shranjevanju so podatki pripravljeni za analize [2].



Slika 3: Arhitektura sistema transformacije podatkov

Tabela 3: Gradniki procesov ETL

Tip objekta	**Opis**
Transformacijski pogled 1	Transformacijski pogled 1 predstavlja srce poslovne rešitve BI4Dynamics, saj je tu skoncentrirana celotna NAV poslovna logika. Na tem pogledu se zgodijo vse transformacije podatkov. Podatki se tu filtrirajo, prečistijo, združijo, sestavijo, preračunajo in predstavijo z granulariteto ciljne dimenzijske tabele ali tabele dejstev. Pri tem se nad podatki lahko uporabi vse funkcije SQL strežnika, kar zagotavlja odprtost platforme BI4Dynamics, za poljubne modifikacije rešitve in razvoj lastnih modulov. Transformacijski pogled 1 bere podatke iz ekvivalentov NAV tabel, nastavitvenih tabel in pomožnih tabel.
Transformacijski pogled 2	Transformacijski pogled 2 je specifičen za tabele dejstev. Značilnost dimenzijskega modeliranja je uporaba t.i. zvezdne sheme, kjer so na tabele dejstev neposredno povezane dimenzijske tabele. V podtakovnem skladišču BI4Dynamics transformacijski pogled 2 predstavlja vezni člen med tabelo dejstev in tabelami dimenzij. Tu se za vsako vrstico tabele dejstev pripnejo šifranti vseh dimenzij, ki v zvezdni shemi to tabelo dejstev obkrožajo.
Polnitvena procedura	Polnitvena procedura je v procesu ETL rešitve BI4Dynamics zadolžena za zapis podatkov v dimenzijske tabele in tabele dejstev. Gre za enostavno proceduro, ki pa se med dimenzijskimi tabelami, tabelami dejstev in pomožnimi tabelami nekoliko razlikuje. Pri tabelah dejstev procedura podatke iz tabele najprej izbriše in nato vse podatke v celoti napolni. V primeru, da se na tabeli izvaja inkrementalno osveževanje, se izbrišejo samo podatki, ki so novejši od inkrementalnega datuma, nastavljenega v nastavitveni tabeli. Dimenzijske tabele se ne brišejo, ker so v zvezdni shemi povezane na tebele dejstev preko tujih ključev. Procedura zato v tabelo shrani samo nove podatke. Ko zaključi s shranjevanjem novih podatkov, pregleda tudi stare in posodobi morebitne spremembe. Polnitvena procedura pomožnih tabel, za razliko od dimenzijskih tabel in tabel dejstev, vsebuje tudi določeno poslovno logiko. V proceduro se lahko prenese del logike, ki bi sicer spadala v transformacijski pogled 1, rezultate pa shrani v pomožno tabelo. Na ta način se lahko močno poenostavi transformacijski pogled 1. Ko polnitvena procedura zaključi s pisanjem podatkov v ciljno tabelo, se trajanje izvajanja in informacija o uspehu shrani v nastavitveno tabelo setup.Log. Z novo arhitekturo se bo pisanje dnevnika iz SQL strežnika prestavilo v .NET klient.

5.2 Prezentacijsko področje

Podatkovno prezentacijsko področje je prostor, kjer so podatki organizirani, shranjeni in neposredno dostopni uporabniškim poizvedbam, sestavljalcem poročil in drugim analitičnim aplikacijam. Sestavljeno je iz skupka integriranih podatkovnih področij, kjer posamezno področje vsebuje podatke enega poslovnega procesa ali funkcije [2].

Za prezentacijsko področje je značilno, da so podatki predstavljeni, shranjeni in dostopni v dimenzijski shemi. Dimenzijsko modeliranje se precej razlikuje od modeliranja s tretjo normalno obliko. Modeliranje s tretjo normalno obliko je načrtovalska tehnika, ki minimizira podatkovno redundanco. Podatki so razdeljeni v številne entitete in vsaka entiteta se v relacijsko bazo preslika kot tabela [2].

Normalizirano modeliranje prispeva veliko k hitri operativni obdelavi podatkov, saj transakcije osveževanja in zapisovanja podatke osvežujejo in zapisujejo v podatkovni bazi samo na enem mestu. Za poizvedbe, ki so tipične za podatkovna skladišča, pa normalizirani modeli niso primerni, saj so strukturno preveč kompleksni. Uporabniki podatkovnega skladišča ponavadi niso dobro seznanjeni s strukturo normaliziranega modela, zato se v njem pri pripravi poizvedb zelo težko znajdejo. Z vidika sistema za upravljanje s podatkovnimi bazami (SUPB) kompleksnost normaliziranih modelov presega zmogljivosti optimizatorja poizvedb, kar predstavlja hud udarec na zmogljivost. Zaradi uporabniških in zmogljivostnih omejitev, uporaba normaliziranega modela v prezentacijskem področju podatkovnega skladišča ni primerna. Podatkovno skladišče, ki ni intuitivno in ne zagotavlja visoke zmogljivosti pri poizvedbah, izgubi namembnost [2].

Pomankljivosti normaliziranih modelov odpravljajo dimenzijski modeli. Ti modeli vsebujejo enake informacije kot normalizirani modeli, podatke pa združijo v obliko, ki zagotavlja enostavnost z vidika uporabnikov, visoko zmogljivost poizvedb in odpornost na spremembe [2].

Posamezna podatkovna področja prezentacijskega področja morajo vsebovati podrobne in atomarne podatke. Zelo pomembno je, da so v strukturi dimenzijskega modela predvidene vse vrste uporabniških poizvedb. Kljub temu da podatkovna področja lahko vsebujejo sumarne podatke, ki izboljšujejo hitrost poizvedb, morajo v modelu obstajati tudi atomarni podatki, ki zagotavljajo najnižjo granularnost. Na ta način lahko uporabnik z gnezdenjem skozi dimenzijske podatke dostopa do najpodrobnejših podatkov. Za prezentacijsko področje je zato zelo pomembno, da je granularnost podatkov tako podrobna in natančna, da ponujajo odgovor na najpodrobnejša in natančna uporabniška vprašanja [2].

Vsa podatkovna področja morajo biti zgrajena iz skupnih dimenzijskih tabel in tabel dejstev. Tem tabelam, ki se pojavljajo v večih podatkovnih področjih, pravimo tudi konformne tabele. Uporaba teh tabel je ključnega pomena za zagotavljanje robustnega in integriranega podatkovnega skladišča. Prezentacijsko področje v večjih podjetjih lahko vsebuje 20 in več podobnih podatkovnih področij. Vsako podatkovno področje vsebuje številne tabele dejstev, vsaka tabela dejstev pa 5 do 15 dimenzijskih tabel. Če je model pravilno načrtovan, so številne dimenzijske tabele skupne večim tabelam dejstev.Arhitekturi, ki vsebuje konformne dimenzijske tabele in tabele dejstev pravimo arhitektura vodila. Taka arhitektura je ključnega pomena pri gradnji porazdeljenih podatkovnih skladišč. V praksi je bolje, da ima analitični sistem centralizirano podatkovno skladišče, vendar v večini podjetij ni dovolj časa, denarja ali politične moči za implementacijo takega sistema[2].

5.2.1 Tabele dejstev

Tabela dejstev je primarna tabela dimenzijskega modela. V njej so shranjene numerične meritve, ki opisujejo zmogljivost poslovnega sistema. Podatki enega poslovnega procesa so ponavadi shranjeni v enem podatkovnem področju. Glede na to, da so meritve daleč največji del kateregakoli podatkovnega področja, se je potrebno izogniti podvajanju meritev med podatkovnimi področji [2].

Termin »dejstvo« opisuje eno poslovno meritev. Če se odločimo, da bomo spremljali poslovanje trgovine, bomo vsak dan za vsak prodan artikel v vseh trgovinah zabeležili meritev o prodani količini in meritev o znesku prodaje. Meritev je zajeta na presečišču vseh dimenzij (dan, artikel in trgovina). Seznam dimenzij definira granularnost tabele dejstev, ki nam pove s kakšnega zornega kota lahko spremljamo meritve [2].

Najbolj uporabne meritve so numerične in aditivne. Aditivnost je zelo pomembna, saj aplikacije, ki izvajajo poizvedbe na podatkovnem skladišču, skoraj nikoli ne vrnejo ene same vrstice iz tabele dejstev. Ponavadi vrnejo stotine, tisoče ali celo milijone vrstic naenkrat pri čemer je najlažje podatke združiti in sešteti. Obstajajo tudi poladitivne meritve po nekaterih dimenzijah ali celo neaditivne meritve, vendar je pri združevanju teh meritvah potrebno uporabiti štetja in računanje povprečij [2].

Vse tabele dejstev imajo dva ali več tujih ključev, ki se povezujejo na primarne ključe dimenzijskih tabel. Ko se vsi ključi tabele dejstev pravilno ujemajo z korespondenčnimi primarnimi ključi dimenzijskih tabel, pravimo, da tabela zagotavlja referenčno integriteto. Do meritev v tabeli dejstev dostopamo preko stičnih dimenzijskih tabel [2].

Sama tabela dejstev ima tudi primarni ključ, ki je sestavljen iz podmnožice tujih ključev. Temu ključu pravimo kompozitni ključ. Vsaka tabela dejstev v dimenzijskem modelu ima kompozitni ključ in vsaka tabela, ki ima kompozitni ključ, je tabela dejstev. To pravilo se v dimenzijskem modelu odraža tako, da je vsaka tabela, ki vsebujejo relacijo mnogo proti mnogo, vedna tabela dejstev. Vse ostale tabele so dimenzijske tabele [2].

5.2.2 Dimenzijske tabele

Dimenzijske tabele so integralni spremljevalci tabel dejstev, ki vsebujejo tekstovne deskriptorje poslovnega sistema. V dobro načrtovanem dimenzijskem modelu imajo dimenzijske tabele veliko opisnih atributov. Pri načrtovanju je potrebno stremeti k vključitvi čim večjega števila smiselnih tekstovnih deskriptorjev. Za dimenzijsko tabelo zato ni nič nenavadnega, da ima 50 do 100 atributov. V primerjavi s tabelami dejstev imajo dimenzijske tabele relativno malo vrstic (pogosto pod milijon vrstic), so pa precej širše zaradi velikega števila atributov. Vsaka dimenzija je definirana z enim stolpcem kot primarnim ključem, ki služi kot osnova za zagotavljanje referenčne integritete z vsemi stičnimi tabelami dejstev [2].

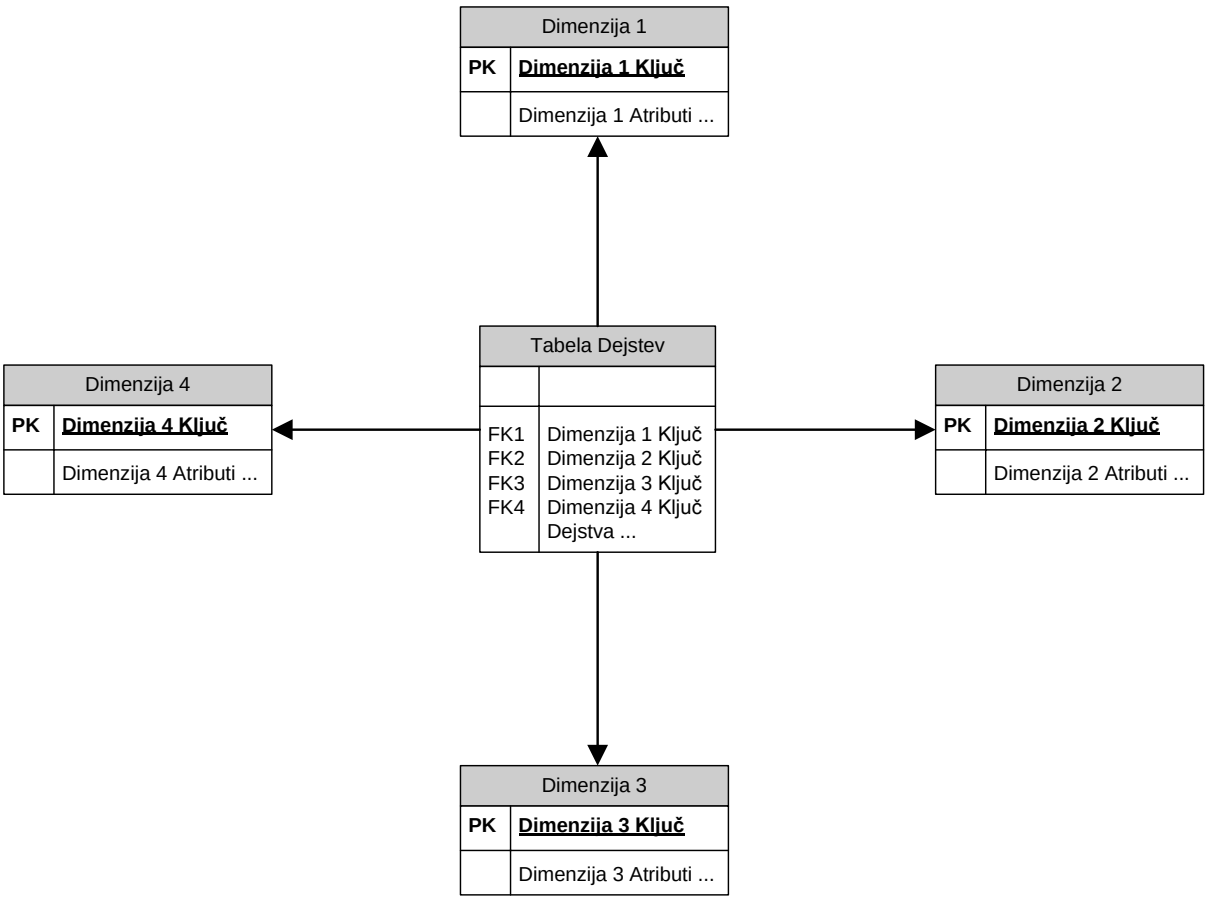
Atributi dimenzijskih tabel igrajo ključno vlogo v podatkovnem skladišču. Glede na to, da so vir vseh zanimivih omejitev in nazivov poročil, zagotavljajo uporabnost in razumevanje podatkovnega skladišča. Pri dimenzijskem načrtovanju je zato zelo pomembno atribute predstaviti z dobro terminologijo, ki kar se da natančno, opisuje poslovni sistem [2].

Najboljši atributi so tekstovni in diskretni. Ne smejo vsebovati okrajšav, ampak morajo biti predstavljeni s celimi besedami. Zato je pri načrtovanju dimenzijskih tabel dobro stremeti k zamenjavi šifrantov s tekstovnimi atributi, ki so bolj opisni in razumljivi [2].

Dimenzijske tabele pogosto vsebujejo hierarhične relacije, ki se pojavljajo v poslovnem sistemu. Če kot primer vzamemo dimenzijsko tabelo artiklov, se bodo verjetno artikli v poslovnem sistemu združili v znamke in znamke naprej v kategorije. Za vsako vrstico v dimenzijski tabeli artiklov je zato potrebno shraniti tudi podatka o znamki in kategoriji. Pri tem se je potrebno zavedati, da so opisni podatki hierarhičnih relacij shranjeni redundančno. Vendar pa je redundanca v tem primeru dobrodošla, saj se na ta način poenostavi uporaba in poviša hitrost poizvedb [2].

5.2.3 Združevanje tabel dejstev in dimenzijskih tabel

Tabela dejstev z numeričnimi meritvami je stično povezana z množico dimenzijskih tabel, ki vsebujejo opisne atribute teh meritev. Tej karakteristični zvezdni strukturi pogosto pravimo tudi zvezdna shema [2].



Slika 4: Zvezdna shema

Prva lastnost, ki se opazi pri zvezdni shemi, je njena enostavnost in simetrija. Očitno je, da poslovni uporabniki z njeno enostavnostjo veliko pridobijo, saj so podatki lažje razumljivi [2].

Enostavnost dimenzijskega modela ima tudi zmogljivnostne prednosti. Optimizatorji podatkovnih baz obdelajo enostavne sheme učinkoviteje z manjšim številom stičnih operacij. SUPB lahko pride do hitrih zaključkov glede ključev močno indeksiranih dimenzijskih tabel, katerim s kartezijskim produktom naenkrat pripne tabelo dejstev in pri tem upošteva vse uporabniške omejitve. Na ta način je možno obdelati vse stične operacije tabele dejstev z enkratnim sprehodom čez njen indeks [2].

Dimenzijski modeli so zaradi svoje enostavnosti in simetričnosti enostavno razširljivi in odporni na spremembe [2].

5.2.4 Prezentacijsko področje BI4Dynamics

Arhitekturno gledano prezentacijsko področje podatkovnega skladišča BI4Dynamics v celoti upošteva primere dobrih praks predstavljene v prejšnjih poglavjih. Podatki prezentacijskega področja so strukturirani in dokončno urejeni. Predstavljeni so v dimenzijski shemi, ki je sestavljena iz šestih podatkovnih področij:

- Področja prodaje
- Področja nabave
- Področja terjatev
- Področja obveznosti
- Področja zalog
- Področja glavne knjige

Podatkovni model podatkovnih področij je realiziran z arhitekturo vodila, kar pomeni, da si področja med seboj delijo skupne dimenzijske tabele. Kljub temu da je podatkovno skladišče BI4Dynamics centralizirano in arhitektura vodila pri tem ne igra tako pomembne vloge, pa njena uporaba znižuje podatkovno redundanco.

Dimenzijske tabele in tabele dejstev so v posameznih podatkovnih področjih strukturirane z zvezdno shemo. Tabele dejstev poleg meritev vsebujejo tudi kompozitne ključe sestavljene iz tujih ključev do stično povezanih dimenzij. Primarni ključ se na tabelah dejstev podatkovnega skladišča BI4Dynamics ne uporablja, ker se podatki med osveževanjem vedno izbrišejo iz tabel in nato ponovno zapišejo. Dimenzijske tabele poleg primarnega ključa vsebujejo veliko opisnih podatkov, ki vključujejo tudi informacije o hierarhičnih relacijah . Pri tem se isti opisni podatek shrani v več različno strukturiranih atributov.

Primer: Opis artikla je predstavljen kot:

- Šifra artikla – ime artikla
- Ime artikla – šifra artikla
- Ime artikla

Z uporabo večih atributov za isti opisni podatek analitični sistem BI4Dynamics poslovnim uporabnikom ponuja večjo izbiro pri gradnji preglednih in uporabnih poročil.

Arhitekture prezentacijskega področja podatkovnega skladišča v novi različici sistema BI4Dynamics ni potrebno spreminjati. Koncept obstoječega dimenzijskega podatkovnega modela in uporaba arhitekture vodila pri načrtovanju posameznih podatkovnih področij zagotavljata modularnost podatkovnega skladišča. V novi arhitekturi se je zato potrebno osredotočiti na modularnost področja priprave podatkov, ki v trenutni arhitekturi predstavlja največji problem.

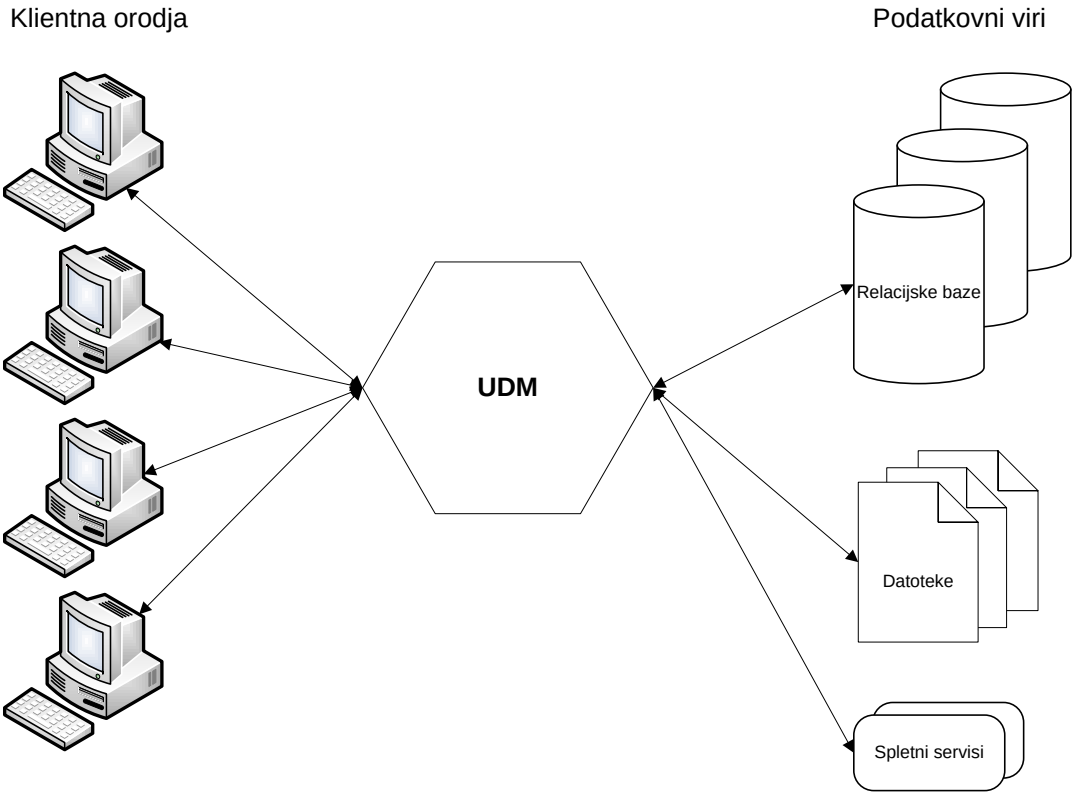
Z novo arhitekturo je pri prezentacijskem področju potrebno poskrbeti za bolj intuitivno in enostavno poimenovanje dimenzijskih tabel in tabel dejstev. Stara poimenovanja so dolga in ne upoštevajo smernic, ki jih predlaga stroka.

- Primer poimenovanja dimenzijske tabele: dwh.dim_Item
- Primer poimenovanja tabele dejstev: dwh.fact_Receivables

Stara arhitektura uporablja shemo »dwh« za označevanje objektov prezentacijskega področja. Predpona »dim_« oz. »fact_« označuje tip posamezne tabele prezentacijskega področja.

5.3 Analitična podatkovna baza

Kljub temu da so podatki na prezentacijskem področju podatkovnega skladišča BI4Dynamcs strukturirani in pripravljeni tako, da so razumljivi in primerni za uporabo s strani poslovnih uporabnikov, pa arhitektura analitičnega sistema med uporabnika in podatke prezentacijskega področja vpeljuje še eno plast- plast analitične baze. Ta plast poslovnim uporabnikom dodatno poenostavi uporabo podatkov, saj seenake poizvedbe na analitični bazi, v primerjavi z relacijskim podatkovnim skladiščem, znatno hitreje izvedejo. Vzrok za to tiči v zelo dobri kompresiji podatkov analitične baze, shranjene na datotečnem sistemu in predpripravljenimi agregati podatkov po različnih nivojih dimenzij. Struktura analitične podatkovne baze je opredeljena v univerzalnem dimenzionalnem modelu (UDM – ang. Universal Dimensional Model).



Slika 5: Univerzalni dimenzionalni model

UDM zagotavlja most med končnimi uporabniki in podatkovnimi viri. Zgrajen je nad enim ali več podatkovnih virov in končnim uporabnikom omogoča, da mu z uporabo klientskih orodij kot je Microsoft Excel, posredujejo poizvedbe.Prednosti, ki jih UDM nudi končnim uporabnikom so [3]:

- berljiv in enostaven podatkovni model
- izolacija od heterogenih zalednih podatkovnih virov
- izboljšana zmogljivost pri agregiranih poizvedbah
- zajem poslovnih pravil v modelu [4]

5.3.1 Objekti analitične podatkovne baze

Analitična baza lahko vsebuje OLAP objekte in objekte za podatkovno rudarjenje kot so:

1. Podatkovni viri
2. Pogledi podatkovnih virov
3. Kocke
4. Meritve
5. Skupine meritev
6. Dimenzije
7. Atribute
8. Hierarhije
9. Strukture za podatkovno rudarjenje
10. Modele za podatkovno rudarjenje
11. Vloge

Poleg teh objektov pa lahko analitična baza vsebuje tudi programske komponente z uporabniško definiranimi funkcijami, ki razširjajo funkcionalnosti jezikov MDX (Multidimensional Expressions) in DMX (Data Mining Extensions) [5].

Analitična podatkovna baza BI4Dynamics vsebuje vse zgoraj naštete tipe objektov, razen vlog ter struktur in modelov za podatkovno rudarjenje. V analitični bazi niso prisotne tudi nobene programske komponente. Vsi ti manjkajoči objekti niso predmet analize obstoječe arhitekture analitične baze BI4Dynamics, zato se v nadaljevanju ne bodo več omenjali.

5.3.2 Podatkovni viri

Podatkovni vir je opredeljen s povezavo do podatkovnega vira. Vsebuje povezovalni niz, ki opisuje način na katerega se analitični strežnik poveže na podatkovni vir. Povezovalni niz vsebuje:

1. ime strežnika
2. ime podatkovne baze
3. varnostni protokol
4. časovno omejitev

Analitični strežnik podpira večje število podatkovnih virov, ki vključujejo tudi podatkovne vire konkurenčnih ponudnikov Oracle, DB2 in Teradata [5].

Relacijska baza podatkovnega skladišča BI4Dynamics je edini vir iz katerega analitična baza črpa podatke. Na podatkovno skladišče se analitični strežnik povezuje preko integriranega SQLNCLI.1 povezovalnika, varna povezava pa je zagotovljena z domensko Windows avtentikacijo.

5.3.3 Pogledi podatkovnih virov

Pogled podatkovnega vira vsebuje logični model sheme, ki jo uporabljajo objekti analitične baze kot so dimenzije in kocke. Predstavljen je z metapodatki, ki so shranjeni v XML strukturi.

Pogled podatkovnega vira:

1. vsebuje metapodatke, ki predstavljajo izbrane objekte enega ali več podatkovnih virov ali metapodatke, ki se bodo uporabili za gradnjo spodaj ležeče podatkovne hrambe.
2. je lahko zgrajen nad enim ali več podatkovnih virov in omogoča uporabniku definicijo OLAP objektov, ki integrirajo podatke iz teh virov.
3. lahko vsebuje relacije, primarne ključe, imena objektov, izračunana polja in poizvedbe, ki niso prisotne v spodaj ležečem podatkovnem viru in obstajajo ločeno od vira.
4. ni viden in dostopen za poizvedbe klientskim aplikacijam [5].

Podatki prezentacijskega področja BI4Dynamics so v pogledu podatkovnega vira na analitično bazo preslikani z 1:1 preslikavo. To pomeni, da pogled ne vsebuje nikakršnih preimenovanj tabel, dodatnih primarnih ključev, izračunanih polj ali dodatnih poizvedb. Pogled je en sam in vsebuje vse dimenzijske tabele in tabele dejstev vseh podatkovnih področij BI4Dynamics.

5.3.4 Kocke

Kocka je množica sorodnih meritev in dimenzij, ki se uporabljajo pri podatkovnih analizah. Posamezna meritev je dejstvo, ki predstavlja transakcijsko vrednost ali meritev po kateri želi uporabnik agregirati podatke. Meritve lahko izhajajo iz stolpcev ene ali več tabel podatkovnega vira. Združene so v skupine meritev [5].

Dimenzija je skupina atributov, ki predstavljajo interesno področje pri analizi sorodnih meritev znotraj kocke. Atributi lahko izhajajo iz stolpcev ene ali več tabel podatkovnega vira in znotraj posamezne dimenzije so lahko organizirani v hierarhije, ki zagotavljajo poti za analize [5].

Kocka je lahko dodatno argumentirana z: [5]

1. Kalkulacijami
2. Ključnimi kazalniki uspešnosti (KPI – angl. Key performance indicator)
3. Akcijami
4. Particijami
5. Perspektivami
6. Prevodi

Kocke analitične baze BI4Dynamics ključnih kazalnikov uspešnosti, akcij in perspektiv ne vsebujejo, zato ti objekti v nadaljevanju ne bodo več obravnavani.

Kalkulacija je skript MDX, ki uporabniku omogoča dodajanje objektov, ki niso definirani s podatki znotraj kocke, ampak z izrazi, ki lahko naslavljajo druge dele kocke, druge kocke ali celo informacije zunaj analitične baze. Kalkulacije omogočajo razširitev zmogljivosti kocke in dodajajo fleksibilnost analitičnim aplikacijam [5].

Številne kocke analitične podatkovne baze BI4Dynamics vsebujejo številne kalkulacije. Tovrste izračunane meritve se uporabljajo predvsem pri izračunu različnih indeksov in

kumulativnih zneskov po posameznih dimenzijah. Tovrstne meritve v kocki niso izračunane vnaprej, ampak se izračunajo v samem analitičnem klientu.

Particije se uporabljajo za shranjevanje podatkov in agregacij skupine meritev kocke analitične baze. Particije so zelo zmogljiv in fleksibilen način, ki omogoča upravljanje posebno velikih kock. Kocka, ki vsebuje zelo veliko prodajnih informacij, lahko za vsako preteklo leto in vsak kvartal tekočega leta vsebuje svojo particijo. Ko se v kocko zapisujejo novi podatki, je potrebno v tem primeru procesirati samo particijo tekočega kvartala. Procesiranje manjših količin podatkov močno znižajo trajanje procesiranja kocke [5]. Pri analitičnem sistemu BI4Dynamics se napredna uporaba particij privzeto ne uporablja. Vsi podatki skupin meritev so shranjeni v eni particiji in se procesirajo vsakič v celoti.

Prevod kocke vsebuje prevode vseh imen objektov in prikaznih datotek znotraj kocke. Prevodi zgotavljajo strežniško podporo večjezičnim klientskim aplikacijam. Podatke pogosto analizirajo uporabniki iz različnih držav, zato je zelo uporabno, da se imena različnih elementov kocke v klientski aplikaciji prikazujejo v lokalnem jeziku uporabnika [5]. Analitični sistem BI4Dynamics podpira večjezičnost.

5.3.5 Dimenzije

Dimenzije so ključni gradnik kock, saj organizirajo podatke v sorodna interesna področja. Dimenzije so skupine atributov, ki temeljijo na stolpcih tabel ali pogledov v pogledu podatkovnega vira. Obstajajo neodvisno od kock. Ista dimenzija se lahko uporabi v več kockah. Prav tako pa je lahko ista dimenzija večkrat vstavljena v isto kocko. Dimenzija, ki obstaja neodvisno od kocke se imenuje dimenzija analitične baze. Instanca dimenzije analitične baze znotraj kocke pa se imenuje dimenzija kocke [5].

Dimenzija analitične podatkovne baze vsebuje atribute, ki se ujemajo s stolpci dimenzijske tabele podatkovnega skladišča. Ti atributi so lahko organizirani v uporabniško definirane hierarhije. Hierarhije se uporabljajo za organizacijo meritev kocke [5].

5.3.6 Namestitev analitične baze BI4Dynamics

Namestitveni čarovnik BI4Dynamics ima celotno analitično bazo z vsemi objekti shranjeno v obliki XMLA (XML for Analysis) ukaza. XMLA je standard, ki omogoča klientskim aplikacijam komunikacijo z več dimenzionalnimi oz. OLAP podatkovnimi viri. Prenos sporočil med aplikacijo in analitičnim strežnikom poteka preko spletnih standardov HTTP, SOAP in XML [6]. Namestitveni čarovnik posreduje ukaz analitičnemu strežniku, ki poskrbi, da se zgradi analitična baza BI4Dynamics z vsemi kockami in dimenzijami.

Tak način namestitve analitične baze je zelo okoren, saj dopušča namestitev samo vnaprej določene strukture objektov. Obstoječa rešitev BI4Dynamics z namestitvenim čarovnikom namesti 6 standarnih kock in tem kockam vse pripadajoče dimenzije.

Kocke:

1. Analiza prodaje
2. Analiza nabave

3. Analiza terjatev
4. Analiza obveznosti
5. Analiza zalog
6. Analiza glavne knjige

Poleg standardnih modulov pa analitični sistem BI4Dynamics ponuja tudi dodatne module.

Dodatni moduli:

1. Kontne preglednice
2. Prodajni naloge
3. Nabavni nalogi

Vseh teh dodatnih modulov z obstoječo arhitekturo ni možno namestiti s pomočjo namestitvenega čarovnika. Za te module je značilno, da niso samostojni moduli, saj se namestijo v že obstoječe standardne kocke. Kontne preglednice tako postanejo del analize glavne knjige, prodajni nalogi del analize prodaje in nabavni nalogi del analize nabave. Z novo arhitekturo je potrebno poskrbeti, da se tudi ti moduli namestijo s pomočjo namestitvenega čarovnika. Pri tem je potrebno zagotoviti tudi mehanizem, ki preverja odvisnosti med moduli. Pri namestitvi modula kontnih preglednic mora mehanizem poskrbeti tudi za namestitev modula glavne knjige. V novi arhitekturi se moduli ne smejo deliti na standardne in dodatne module. Vsi moduli so enakovredni in namestijo se lahko povsem samostojno.

Z novo arihtekturo je poleg modularnosti kock potrebno zagotoviti tudi modularnost dimenzij. V obstoječi arhitekturi je nabor dimenzij predefiniran v XMLA ukazu, tako da se vedno namestijo vse dimenzije. Nova arhitektura BI4Dynamics mora predviditi namestitev samo potrebnih dimenzij. Z vpeljavo poljubnega nameščanja modulov kock je potrebno v novi arhitekturi implementirati mehanizem, ki bo preverjal odvisnosti kock in dimenzij. Pri namestitvi modula poljubne kocke mora mehanizem zagotoviti predhodno namestitev vseh dimenzij, ki v kocki nastopajo. Namestitveni čarovnik mora za vsak analitični modul uporabniku omogočati izklop poljubnih dimenzij. Na ta način si uporabnik v kocki prikaže samo podatke, ki so potrebni za analize.

6 Arhitektura podatkovnega modela BI4Dynamics 3.0

6.1 Modularno ogrodje

Prenovljena arhitektura podatkovnega modela BI4Dynamics 2.4 mora na nivoju podatkovnega skladišča zagotoviti modularno izgradnjo področja priprave podatkov in prezentacijskega področja. Na nivoju analitične baze mora arhitektura ukiniti namestitveni dokument XMLA in posamezne dimenzije ter kocke analitične baze predstaviti s samostojnimiin v XML obliko zapisanimiobjekti.

Stara arhitektura je modularnost prezentacijskega področja že zagotavljala na račun značilnosti dimenzijskega modela podatkovnega skladišča, ki je že po definiciji zasnovan modularno. Problem se je pojavil pri komformnih dimenzijskih objektih in komformnih objektih dejstev, kjer je bilo potrebno zagotoviti pravilni vrstni red namestitve teh objektov. Tako so se morali namestiti vsi dimenzijski objekti pred namestitvijo objektov dejstev, v katerih je vključena dimenzija.

Stara arhitektura se je izognila vsem tem problemom z odvisnostmi med moduli z delitvijo modulov na standardne in dodatne module. Vsaka instanca analitičnega sistema BI4Dynamics 2.4 vsebuje vedno vsaj 6 standardnih modulov, ki zagotavljajo vse potrebne objekte za namestitev ostalih dodatnih modulov. Za medsebojne odvisnosti objektov standardnih modulov je bilo poskrbljeno s pravilnim razvrščanjem objektov v seznam, iz katerega je namestitveni čarovnik dobil objekte v pravilnem vrstnem redu.

Z novo arhitekturo se zaradi komercialnih razlogov ukinja delitev modulov na standardne in dodatne module. Vsi moduli so v novi arhitekturi povsem enakovredni, kar pomeni, da lahko vsaka instanca analitičnega sistema BI4Dynamics 3.0 vsebuje poljubno število modulov. S prenovo arhitekture na tak način pa se izgubi informacija o odvisnostih med moduli. Namestitveni čarovnik analitičnega sistema BI4Dynamics 3.0 mora zato zagotoviti mehanizem za uporavljanje odvisnosti med moduli.

Zelo pomembna novost, ki jo prinaša nova arhitektura BI4Dynamics, je podpora različicam objektov podatkovnega skladišča in objektov analitične baze BI4Dynamics. Za vsak objekt modula arhitektura omogoča namestitev poljubne različice objekta glede na predefinirane parametre:

1. Različica analitičnega sistema BI4Dynamics
2. Različica podatkovnega vira
3. Različica SQL Strežnika
4. Tip SQL Strežnika

Stara arhitektura je imela za vse objekte skripte SQL prilagojene optimalnemu delovanju na standardni različici SQL Strežnika 2005 na podatkovnem modelu transakcijskega sistema NAV 4.0. Ta pomanjkljivost pa je pomenila veliko omejitev pri optimizaciji procesov ETL, ki se izvajajo pri pripravi podatkov. S prihodom novejših različic transakcijskega sistema NAV, je Microsoft poskrbel za številne spremembe na podatovnem modelu, ki omogočajo dostop do nekaterih podatkov preko manjšega števila stičnih operacij, ki se uporabljajo v

transformacijskih pogledih procesov ETL. Poleg izboljšav na samem transakcijskem sistemu so pri Microsoftu poskrbeli za številne nove zmogljivosti SQL strežnika s prihodom različice 2008.

Podpora različicam objektov BI4Dynamics prinaša možnost uporabe najnovejših konceptov, postopkov in zmogljivosti, ki jih narekuje razvoj novih različic transakcijskega vira NAV in SQL strežnika.

6.1.1 Modul

V modul prenovljene arhitekture BI4Dynamics se zapakirajo 3 tipi komponent podatkovnega modela:

1. Modul ogrodja BI4Dynamcs
2. Modul dimenzije
3. Modul kocke

Modul ogrodja BI4Dynamics vsebuje vse nastavitvene tabele in nastavitvene procedure, ki izgradijo, napolnijo in indeksirajo področje priprave podatkov. Poleg nastavitvenih procedur, pa sta v modulu tudi proceduri za izbris poljubnega objekta podatkovnega skladišča in procedura za inicializacijo dimenzijske tabele.

Modula dimenzije in kocke sta strukturno zelo podobna. Vsebujeta:

1. Nastavitveni dokument XML tabel in stolpcev tabel vira, iz katerih modul črpa podatke
2. V samostojne dokumente XML zapakirane vse skripte SQL modula
3. Strukturni dokument XML objekta na analitični bazi
4. Dokument XML z vrstnim redom nameščanja posameznih komponent modula
5. Dokument XML z vrstnim redom polnjenja tabel modula

```
<?xml version="1.0" encoding="utf-16"?>
<objects>
    <object BI4DynamicsVersion = "" DataSourceVersion = "" SqlVersion = "" SqlType = "">
        <tables>
            <table>
                <tableID>95</tableID>
                <tableName>G/L Budget Name</tableName>
                <stageIndexes>
                    <stageIndex>
                        <indexName>IX_GLBudgetName</indexName>
                        <indexColumn>[Name]</indexColumn>
                        <clusteredIndex>true</clusteredIndex>
                    </stageIndex>
                </stageIndexes>
                <columns>
                    <column>
                        <columnID>1</columnID>
                        <columnName>Name</columnName>
                    </column>
                    <column>
                        <columnID>2</columnID>
                        <columnName>Description</columnName>
                    </column>
                </columns>
            </table>
        </tables>
    </object>
</objects>
```

Slika 6: Primer nastavitvenega dokumenta XML tabel in stolpcev vira

```
<?xml version="1.0" encoding="utf-16"?>
<objects>
    <object BI4DynamicsVersion = "" DataSourceVersion = "" SqlVersion = "" SqlType = "">
        -- ############################################################################## --
        -- #####                        View: dim.GLBudgetView                      ##### --
        -- ############################################################################## --
        EXEC dbo.DropObject 'dim.GLBudgetView', 'V'
        GO
        CREATE VIEW dim.GLBudgetView AS
            SELECT
                a.Name,
                Description = ISNULL(NULLIF(a.Description, ''), 'N/A'),
                NameDesc = a.Name + ISNULL(' - ' + NULLIF(a.Description, ''), ''),
                a.CompanyID,
                a.DataSourceID
            FROM
                stage.GLBudgetName a
        GO
    </object>
</objects>
```

Slika 7: Primer zapakirane skripte SQL v dokument XML

```
<?xml version="1.0" encoding="utf-16"?>
<objects>
    <object BI4DynamicsVersion = "" DataSourceVersion = "" SqlVersion = "" SqlType = "">
        <deploymentFlow>
            <deployStage>
                <deployObject>TablesColumns.xml</deployObject>
            </deployStage>
      <deployDwh>
                <deployObject>dim.GLBudgetView.xml</deployObject>
                <deployObject>dim.GLBudget.xml</deployObject>
                <deployObject>dim.LoadGLBudget.xml</deployObject>
            </deployDwh>
      <deployUdm>
        <deployObject>GLBudget.xml</deployObject>
        <dswTables>
          <deployObject>dim.GLBudget</deployObject>
        </dswTables>
            </deployUdm>
        </deploymentFlow>
    </object>
</objects>
```

Slika 8: Primer poteka nameščanja modula v dokumentu XML

Kot je iz primerov dokumentov XML razvidno, je znotraj vsakega dokumenta poskrbljeno za podporo večim različicam istega objekta. Za izbiro ustrezne različice objekta pri nameščanju skrbi mehanizem znotraj ogrodja .NET, ki se mu reče ogrodje MEF (angl. Module Extensibility Framework).

6.1.2 Ogrodje MEF

MEF je ogrodje, ki poenostavlja gradnjo razširljivih aplikacij v okolju .NET tako, da omogoča enostavno nalaganje razširitvenih modulov aplikacije. Do prihoda ogrodja MEF je bilo potrebno v aplikacijah, ki so podpirale model vtičnikov, postaviti svojo infrastrukturo za ta problem. Posledica je bila, da so bili vtičniki odvisni od posamezne aplikacije, kar je onemogočalo ponovno uporabo vtičnika v različnih implementacijah [7].

MEF zagotavlja standardno rešitev, ki aplikacijam omogoča uporabo zunanjih vtičnikov, ki se lahko ponovno uporabijo v številnih drugih aplikacijah. Nekateri vtičniki so lahko tudi z uporabo ogrodja MEF implementirani na način, ki je specifičen samo za določen tip aplikacije. Med posameznimi vtičniki aplikacije lahko obstajajo medsebojne odvisnosti, za katere poskrbi MEF, tako da zagotavlja povezovanje vtičnikov v pravilnem vrstnem redu. MEF omogoča tudi različne pristope, po katerih lahko aplikacije najdejo in naložijo vtičnik. Pri tem si pomagajo s posebnimi metapodatki in oznakami, s katerimi MEF omogoča široke možnosti pri iskanju in filtriranju vtičnikov [7].

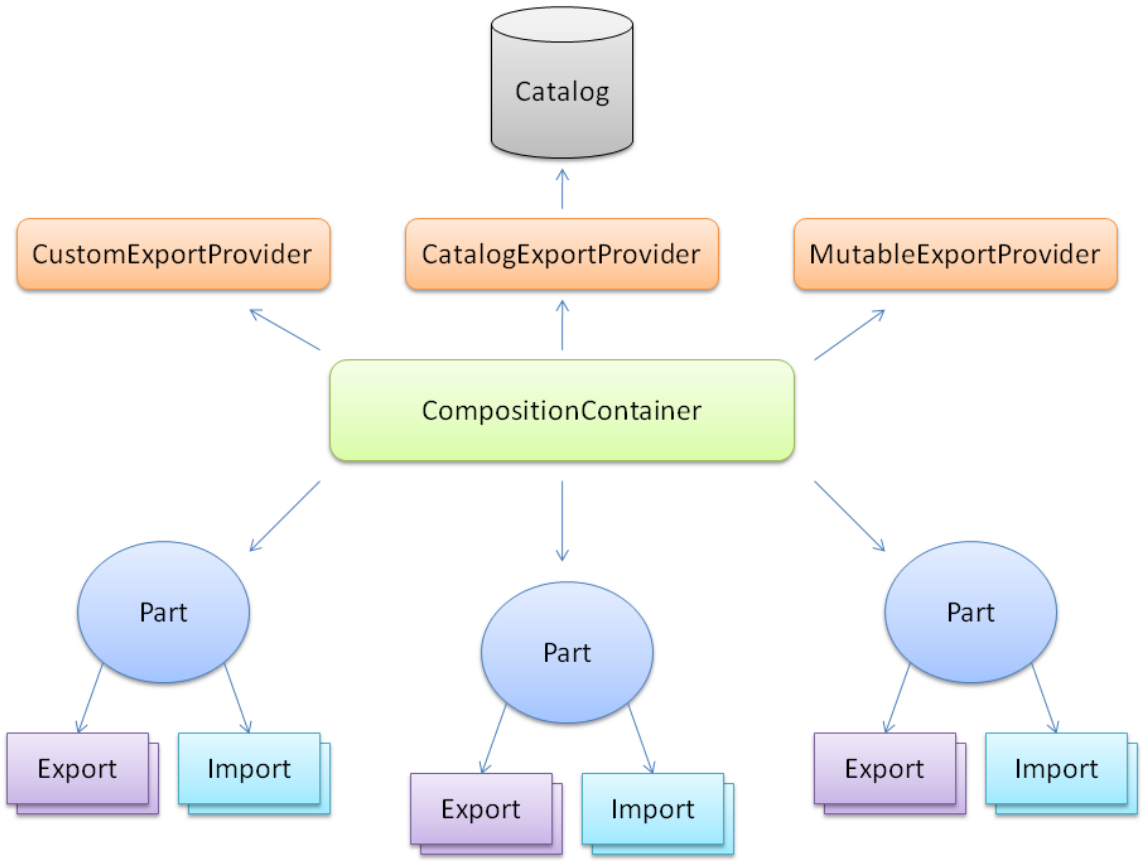
Jedro ogrodja MEF sestavljata katalog in vsebnik sestavnih delov, kjer katalog omogoča odkrivanje vtičnikov, vsebnik pa koordinira kreacijo in podporo odvisnosti vtičnikov.

Najpomembnejši del ogrodja MEF je sestavni del, ki ima lahko enega ali več izvozov in je lahko odvisen od ene ali več zunanjih storitev oz. uvozov. Instanca sestavnega dela je objekt poljubnega tipa. MEF je v osnovi razširljiv in omogoča dodajanje dodatnih implementacij sestavnih delov, dokler se sestavni deli držijo uvoznih/izvoznih pogodb. Vsak uvoz in izvoz sta povezana preko pogodbe. Pogodba vsebuje metapodatke, ki se lahko uporabijo za filtriranje in iskanje vtičnika [7].

MEF vsebnik je v interakciji s katalogi, preko katarih ima dostop do vseh sestavnih delov. Vsebnik sam razreši nekatere odvisnosti in poskrbi, da se izvozi izpostavijo aplikacijam. Instance sestavnih delov se lahko doda neposredno na vsebnik sestavnih delov.

Sestavni del, ki ga vrne katalog, je ponavadi kar vtičnik za posamezno aplikacijo. Lahko vsebuje uvoze (odvisnosti) in izvoze komponent, ki jih ciljna aplikacija ponuja ali izvaža. [7]

Privzeta implementacija sestavnega dela v ogrodju MEF uporablja metapodatke, s katerimi so definirani vsi izvozi in uvozi. To omogoča ogrodju MEF, da določi sestavne dele, uvoze in izvoze, ki so dostopni v sami aplikaciji [7].



Slika 9: Shema ogrodja MEF

Nova različica analitičnega sistema BI4Dynamics ogrodje MEF uporablja za implementacijo analitičnih modulov kot medsebojno odvisne vtičnike, ki se namestijo s pomočjo .NET klienta v vrstnem redu, ki ga narekujejo njihove odvisnosti. Iz kataloga knjižnic DLL, ki sestavljajo aplikacijo BI4Dynamics, se iz imenskih prostorov izluščijo objekti in procedure, ki so označeni za izvoz, in formira se vsebnik razširljivih objektov in procedur. Te razširljive objekte in procedure je možno priklicati kjerkoli znotraj imenskih prostorov aplikacije.

6.2 Področje priprave podatkov

Področje priprave podatkov stare različice podatkovnega skladišča BI4Dynamics je strukturno zelo okorno in neprilagodljivo. Za zagotavljanje popolne modularnosti celotnega analitičnega sistema BI4Dynamics je taka arhitektura neprimerna, zato jo je potrebno posodobiti in poenostaviti do te mere, da bo tudi samo področje priprave podatkov povsem modularno. Posodobitve stare arhitekture, ki to zagotavljajo so:

1. Zmanjšanje števila nastavitvenih tabel
2. Zapisovanje nastavitev v nastavitvene tabele neposredno iz vira
3. Razbitje enega nastavitvenega dokumenta XML tabel in stolpcev v več dokumentov, ki za vsak modul vsebujejo samo modulu potrebne tabele in stolpce
4. Odprava modulom namenskih pomožnih tabel za prevode opcijskih polj

5. Izgradnja, polnitev in indeksiranje celotnega področja priprave podatkov se izvede zaporedno v samo treh korakih
6. Postavitev mehanizma, ki omogoča postavitev poljubnega števila indeksov na poljubnem številu tabel na poljubni kombinaciji stolpcev tabele.

6.2.1 Nastavitvene tabele

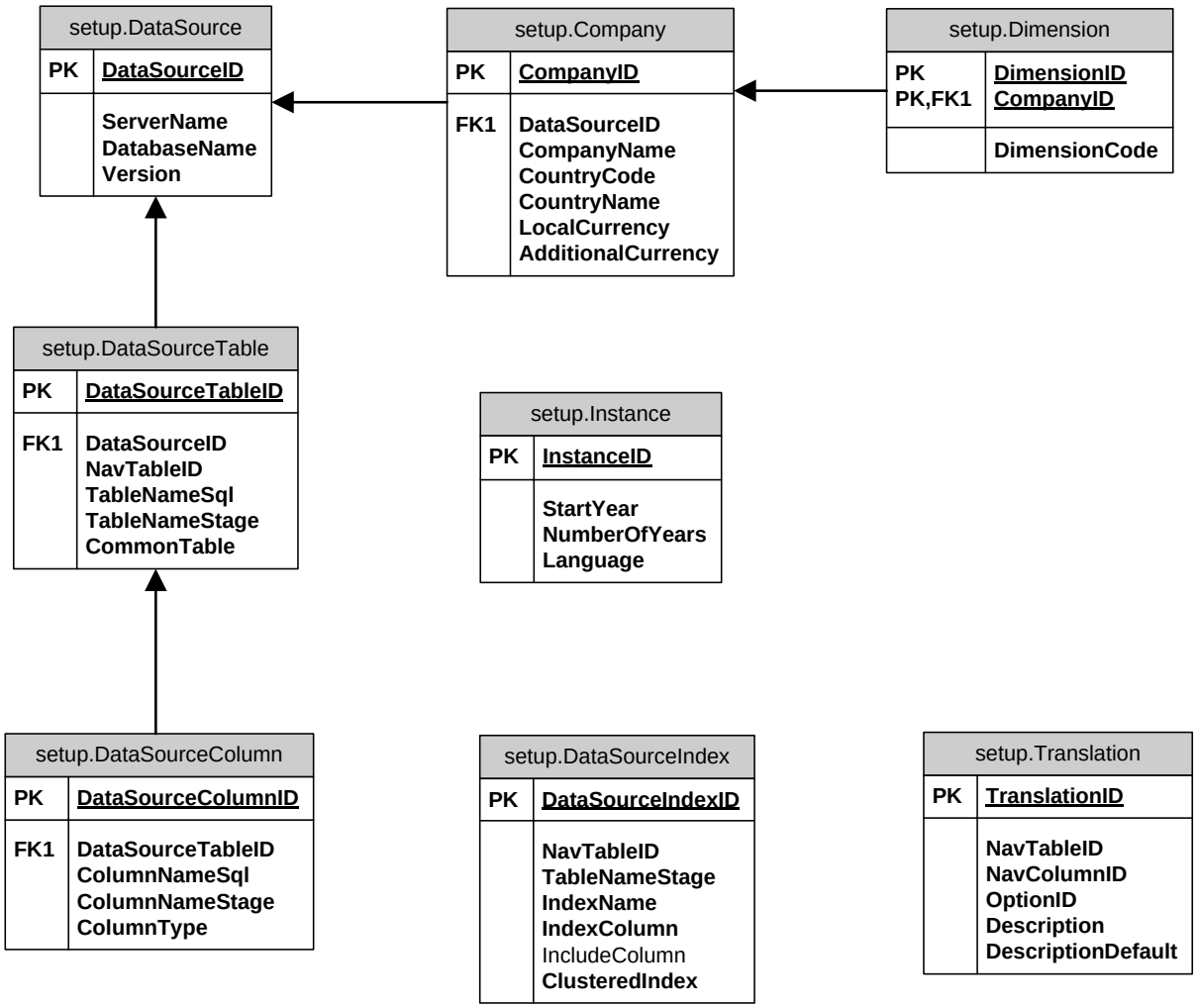
Nova arhitektura področja priprave podatkov strukturno spreminja koncept nastavitvenih tabel, ne spreminja pa njihove namembnosti. Funkcija tabel je še zmeraj hranjenje nastavitvenih podatkov virov in podatkov, ki so vezani na uporabniško definirane nastavitve celotnega analitičnega sistema BI4Dynamics.

Strukturne spremembe na področju nastavitvenih tabel, ki jih vpeljuje nova arhitektura so načrtovane s smernicami za:

1. Minimizacijo nabora nastavitev vira in analitičnega sistema BI4Dynamics
2. Odpravo vseh obsolitskih strukturnih elementov nastavitvenih tabel
3. Minimizacijo strukturne odvisnosti nastavitvenih tabel s strukturo izvornih transakcijskih sistemov

Minimizacija nabora nastavitev in odbrava vseh obsolitskih strukturnih elementov poenostavljata koncept področja priprave podatkov in predstavljata temelje za poenostavitev izgradnje duplikatov tabel NAV.

Z minimizacijo strukturne odvisnosti nastavitvenih tabel od samega izvornega transakcijskega sistema NAV nova arhitektura vpeljuje možnost enostavne razširitve analitičnega sistema BI4Dynamics tudi na druge transakcijske sisteme kot sta Microsoft Dynamics AX in Microsoft Dynamics CRM z minimalnimi strukturnimi spremembami nastavitvenih tabel.



Slika 10: Shema nastavitvenih tabel v novi arhitekturi

Tabela 4: Nastavitvene tabele novega področja priprave podatkov

Tabela	**Opis**
setup.DataSource	Tabela je ekvivalent tabeli setup.BasicSetting v stari arhitekturi. Z novo arhitekturo poleg prijaznejšega imena doživela tudi poenostavitev. Tabela za vsak podatkovni vir vsebuje naslednje podatke: • Ime strežnika vira • Ime podatkovne baze vira • Različico vira Za vnos podatkovnih virov v tabelo skrbi .NET klient. Klient omogoča vnos poljubnega števila podatkovnih virov.
setup.DataSourceColumn	Tabela združuje tabele setup.OriginalColumnList in setup.ColumnList stare arhitekture. V tabeli so shranjeni podatki o vseh stolpcih vseh duplikatov tabel NAV iz vseh podatkovnih virov, ki se nahajajo na področju priprave podatkov podatkovnega skladišča BI4Dynamics. Tabela za vsak stolpec vsebuje naslednje podatek o:

	• pripradnosti tabeli • SQL imenu stolpca na NAV podatkovnem viru • imenu stolpca v duplikatu NAV tabele • podatkovnem tipu stolpca Tabela je načrtovana tako, da za vsak stolpec vsebuje toliko vrstic, kot je virov vključenih v podatkovno skladišče. Tako podrobna predstavitev stolpcev je potrebna, ker se strukture različnih različic NAV podatkovnih virov med seboj lahko razlikujejo. Določeni stolpci so tako prisotni samo v določenih različicah NAV. Nekateri stolpci, ki so vsebinsko sicer prisotni v vseh različicah NAV, pa se lahko od različice do različice razlikujejo v poimenovanju. Duplikat vsake NAV tabele na področju priprave podatkov tako vsebuje unijo vseh stolpcev tabele po vseh podatkovnih virih. Procedura, ki polni duplikate NAV tabel, v tako strukturirani tabeli za vsak podatkovni vir dobi informacije o vseh stolpcih posamezne tabele.
setup.Company	Tabela setup.Company v novi arhitekturi ni doživela večjih sprememb. Za vsako podjetje tabela vsebuje podatek o: • podatkovnem viru, kateremu pripada podjetje • imenu podjetja • oznaki države podjetja • imenu države podjetja • lokalni valuti podjetja • konsolidacijski valuti podjetja Nova tabela ne vsebuje več podatka o drugi konsolidacijski valuti.
setup.Dimension	Tabela je ekvivalent tabeli setup.GlobalDimension v stari arhitekturi. Z novo arhitekturo je dobila novo ime. Tabela za vsako dimenzijo vsebuje naslednje podatke: • Šifro podjetja, kateremu dimenzija pripada • Oznako dimenzije
setup.DataSourceIndex	Stara arhitektura je za hitrejše izvajanje transformacijskih pogledov na tabelah področja priprave podatkov poskrbela z indeksi na stolpcih primarnega ključa tabele in dodatnimi indeksi na stolpcih, preko katerih se stično združuje več tabel. Indeksiranje primarnega ključa je potekalo dinamično s pomočjo datoteke XML, ki je poleg informacij o tabelah in stolpcih, vsebovala tudi informacijo o primarnem ključu NAV tabele. Na številnih tabelah področja priprave podatkov je stara arhitektura poskrbela za postavitev indeksov tudi na poljih, ki niso vsebovana v primarnem ključu. Seznam teh indeksov je bil definiran znotraj namenske procedure, ki pa je bila povsem statična, brez možnosti dodajanja dodatnih indeksov ali pa dodajanja novih polj v že obstoječe indekse. Z novo arhitekturo je namen poenotiti gradnjo indeksov na vseh tabelah področja priprave podatkov, pri tem pa, pri razvoju modulov, omogočiti popoln nadzor nad indeksiranjem tabel. V

	novi arhitekturi BI4Dynamics je tako na poljubni tabeli področja priprave podatkov možno postaviti poljubno število različnih indeksov (clustered ali nonclustered) in pri tem v vsak indeks vključiti poljubno kombinacijo polj izvorne NAV tabele. Vsak modul v novi arhitekturi BI4Dynamics ima svoj nastavitveni dokument XML z vsemi tabelami NAV in njihovimi stolpci, ki so potrebni za pravilno namestitev modula. Poleg teh informacij nastavitveni dokument XML vsebuje tudi informacije o vseh morebitnih indeksih na ekvivalentu NAV tabel na področju priprave podatkov. Te informacije se pred namestitvijo modula zapišejo v nastavitveno tabelo setup.DataSourceIndex. Podatki, ki jih nastavitvena tabela vsebuje so: • Šifra tabele NAV • Ime tabele v področju priprave podatkov • Ime indeksa • Seznam polj za indeksiranje • Tip indeksa (clustered, non-clustered) • Stolpci, ki se pripnejo indeksu tipa non-clustered Nastavitvena tabela služi kot vir podatkov za nastavitveno proceduro, ki poskrbi, da se na tabelah področja priprave podatkov postavijo vsi potrebni indeksi.
setup.DataSourceTable	Tabela združuje tabele setup.OriginalTableList in setup.TableList stare arhitekture. V tabeli so shranjeni podatki o vseh duplikatih tabel NAV iz vseh podatkovnih virov, ki se nahajajo na področju priprave podatkov podatkovnega skladišča BI4Dynamics. Tabela za vsako tabelo vsebuje podatke o: • pripradnosti podatkovnemu viru • šifri tabele na podatkovnem viru • SQL imenu tabele na NAV podatkovnem viru • imenu duplikata NAV tabele • skupni tabeli Tabela je načrtovana tako, da za vsako tabelo vsebuje toliko vrstic, kot je virov vključenih v podatkovno skladišče. Tako podrobna predstavitev tabel je potrebna, ker se strukture različnih različic NAV podatkovnih virov med seboj lahko razlikujejo. Določene tabele so tako prisotne samo v določenih različicah NAV. Nekatere tabele, ki so vsebinsko sicer prisotne v vseh različicah NAV, pa se lahko od različice do različice razlikujejo v poimenovanju. Procedure, ki kreirajo, polnijo in indeksirajo duplikate NAV tabel, v tako strukturirani tabeli za vsak podatkovni vir dobijo informacije o vseh tabelah posameznega vira.
setup.Instance	Tabela je ekvivalent tabeli setup.Property v stari arhitekturi. Z novo arhitekturo je dobila novo ime. V vsaki instanci podatkovnega skladišča BI4Dynamics tabela vsebuje vrstico z naslednjimi podatki: • Začetni datum podatkov • Število let v prihodnosti, za katera v NAV obstajajo knjižbe

	• Jezik predstavitve podatkov Tabela ima v primerjavi s predhodnico nekoliko enostavnejšo strukturo z manj podatki. Nova arhitektura ukinja funkcionalnost inkrementalnega osveževanja podatkovnega skladišča, zato nastavitvena polja za inkrementalno osveževanje več niso potrebna.
setup.Translation	Velika večina tabel NAV vsebuje številna opcijska polja. Ta polja vsebujejo nabor predefiniranih vrednosti, ki jih avtomatika ali poslovni uporabnik sam nastavi pri zajemanju podatkov. Primeri takih polj so: tip dokumenta, tip zapiranja, tip postavke, … Za opcijska polja tabel NAV je značilno, da so na nivoju podatkovne baze predstavljena s številčnim enumeratorjem. V NAV klientu pa je integrirana logika, ki poskrbi za preslikavo teh številčnih enumeratorjev v opisne vrednosti. Na ta način se na nivoju podatkovne baze prihrani veliko prostora, uporabniku pa omogoči prijazen prikaz vrednosti opcijskega polja. Z vidika podatkovnega skladišča BI4Dynamics pa integrirana logika, ki skrbi za preslikavo enumeratorjev v opise vrednosti, predstavlja velik problem za uporabniku prijazen prikaz vrednosti opcijskih polj. Vrednosti se ne nahajajo nikjer na podatkovni bazi NAV od koder nastavitveni čarovnik BI4Dynamics bere vse potrebne podatke. Čarovnik nima neposrednega dostopa do NAV logike, ki skrbi za povezavo med numerično in opisno predstavitvijo vrednosti opcijskih polj. V stari arhitekturi se je problem reševal z svojo nastavitveno tabelo za vsak opcijski atribut, ki se uporablja v modulih BI4Dynamics. Teh atributov je v modulih precej, zato nova arhitektura predvideva poenostavitev množice pomožnih tabel na eno. Tabela se iz nabora pomožnih tabel prestavi v nabor nastavitvenih tabel. Nastavitvena tabela, ki je namenjena povezovanju numerične predstavitve opcijskega polja na viru in opisne predstavitve opcijskega polja v podatkovnem skladišču in dimenzijah na analitičnem strežniku, vsebuje naslednje podatke: • Šifro jezika • Šifro tabele vira • Šifro stolpca znotraj tabele vira • Numerično vrednost enumeratorja opcijskega polja • Prevod opisne vrednosti opcijskega polja • Privzeto opisno vrednost opcijskega polja

6.2.2 Izgradnja področja priprave podatkov

Proces izgradnje področja priprave podatkov nova arhitektura analitičnega sistema BI4Dynamics močno poenostavlja. Pri tem igrata vključno vlogo ukinitev podpore Native različici transakcijskega sistema NAV in opuščanje inkrementalnega osveževanja podatkov področja priprave podatkov.

Poenostavitev izgradnje področja priprave podatkov se kaže v:

1. Opuščanju polnitvenih procedur za vsak ekvivalent tabele NAV

2. Polnitev vseh tabel s pomočjo ene procedure, ki kopira podatke iz vseh izvornih transakcijskih sistemov v enem obhodu.
3. Indeksiranje tabel s pomočjo nastavitev v indeksom namenski nastavitveni tabeli
4. Prenos izvajanja kreacije, polnitve in indeksiranja tabel iz strežnika SQL na BI4Dynamics klientsko aplikacijo.

Slika 11: Namestitev področja za pripravo podatkov

Tabela 5: Koraki namestitve novega področja priprave podatkov

Aktivnost	**Opis**
Kreiraj proceduro za izbris objektov	Pri namestitvi področja priprave podatkov se tudi v novi arhitekturi BI4Dynamics najprej namesti procedure dbo.DropObject. S pomočjo te procedure se lahko izbriše vse objekte podatkovnega skladišča. Z novo arhitekturo procedura ni doživela nobenih sprememb.
Kreiraj nastavitvene tabele	Nastavitvene tabele se tudi v novi arhitekturi namestijo kot del področja priprave podatkov. V njih se zapisujejo informacije o določenih NAV nastavitvah in vse potrebne nastavitve za izgradnjo podatkovnega skladišča BI4Dynamics. Z novo arhitekturo se je število nastavitvenih tabel zmanjšalo in večina tabel je dobila prijaznejša imena. Nova različica BI4Dynamics vsebuje naslednje nastavitvene tabele: • setup.DataSource • setup.Company • setup.Instance • setup.Dimension • setup.DataSourceTable • setup.DataSourceColumn • setup.DataSourceIndex
Kreiraj inicializacijsko proceduro dimenzij	Procedura setup.InitDimension je z novo arhitekturo doživela manjšo spremembo. Še vedno je njena naloga, da ob kreaciji dimenzijskih tabel poskrbi za vnos posebne vrstice v novo ustvarjeno dimenzijo, ki ima v vseh dimenzijah šifrant 0 in za vsa polja dimenzije vsebuje privzeto vrednost odvisno od podatkovnega tipa polja. Razlikuje pa se v načinu določanja privzetih vrednostih. Stara procedura je privzeto vrednost za posamezen podatkovni tip prebrala v nastavitveni tabeli setup.DefaultDimensionValue. V novi arhitekturi ta nastavitvena tabela ni več predvidena, zato so privzete vrednosti podatkovnih tipov shranjene kot del procedure.
Kreiraj tabele področja priprave podatkov	Ekvivalenti tabel NAV na področju priprave podatkov se v novi različici BI4Dynamics zgradijo s pomočjo procedure setup.CreateStage. Procedura za vsako tabelo NAV zapisano v nastavitveni tabeli setup.DataSourceTable gradi CREATE skript, ki kreira ekvivalent tabele na področju priprave podatkov. Stolpce in njihove podatkovne tipe procedura za vsako tabelo prebere v nastavitveni tabeli setup.DataSourceColumn. Seznam stolpcev se tako kot v predhodnici procedure sestavlja dinamično in na konec seznama se dodata še šifranta CompanyID in DataSourceID.
Napolni podatke v področje priprave podatkov	V novi arhitekturi so tabele področja priprave podatkov izgubile svoje lastne polnitvene procedure, ki so vanje prenašale podatke iz virov. S tem se je koncept prenosa podatkov v področje priprave podatkov močno poenostavil. Podatki se v novi različici BI4Dynamics iz NAV virov v področje priprave podatkov prenašajo v vse tabele naenkrat znotraj

	procedure setup.LoadStage. Procedura za vsako tabelo iz nastavitvene tabele setup.DataSourceTable gradi INSERT ukaz, ki iz vseh podatkovnih virov kopira podatke v ekvivalent tabele na področju priprave podatkov. Seznam stolpcev ciljne tabele področja priprave podatkov dobi procedura v polju ColumnNameStage nastavitvene tabele setup.DataSourceColumn. Zajem podatkov iz različnih podatkovnih virov je realiziran z uporabo operacije UNION nad SELECT poizvedbami na tabeli NAV.
Indeksiraj področje priprave podatkov	Indeksiranje tabel področja priprave podatkov v novi arhitekturi ni več razdeljeno na indeksiranje stolpcev primarnega ključa tabele NAV in nameščanje posebnih indeksov. Z novo arhitekturo je vpeljan enoten sistem, ki za poljubno tabelo NAV na področju priprave podatkov postavi poljubno število indeksov na poljubni kombinaciji stolpcev. V novi različici BI4Dynamics je vpeljana nova nastavitvena tabela setup.DataSourceIndex, ki vsebuje seznam vseh indeksov na tabelah področja priprave podatkov. Procedura setup.IndexStage gradi CREATE CLUSTERED INDEX oz.CREATE NONCLUSTERED INDEX ukaz, pri čemer seznam stolpcev indeksa, tip indeksa in morebitne pripete stolpce prebere iz nastavitvene tabele. Vsi indeksi se zgradijo v enem obhodu procedure.

6.3 Prezentacijsko področje

Prezentacijsko področje podatkovnega skladišča s prenovo podatkovnega modela ni doživelo strukturnih sprememb, saj je dimenzijski model že v stari arhitekturi zagotavljal modularno strukturo podatkovnega modela.

Z novo arhitekturo je prezentacijsko področje doživelo nekaj lepotnih popravkov z enostavnejšim preimenovanjem objektov podatkovnega skladišča. Stara arhitektura je objekte prezentacijskega področja vsebovala na shemi »dwh«., tip objekta (dimenzijska tabela ali tabela dejstev) pa je bil podan na začetku imena objekta.

Primer starega poimenovanja:

dwh.dim_Company ali dwh.fact_Receivables

Nova arhitektura ukinja shemo »dwh« in za celotno podatkovno skladišče vpeljuje shemi »dim« in »fact«. Na ta način se poenostavi poimenovanje objektov podatkovnega skladišča in poveča preglednost med seboj odvisnih elementov v orodju SQL Management Studio, saj so, poleg dimenzijskih tabel in tabel dejstev, na ti dve shemi pripeti tudi vsi objekti procesov ETL (transformacijski pogledi, polnitvene procedure) področja priprave podatkov. Vsi med seboj odvisni objekti so razporejeni po abecednem vrstnem redu eden pod drugim. Na ta način se poenostavi razvoj dodelav na podatkovnem skladišču, saj se močno olajša iskanje sorodnih objektov.

Primer novega poimenovanja:

dim.Company ali fact.Receivables

6.4 Analitična podatkovna baza

Zadnje področje, ki je s prenovo arhitekture podatkovnega modela doživelo večje spremembe je analitična podatkovna baza BI4Dynamics. Podatkovni model se v novi arhitekturi vsebinsko in strukturno praktično ni spremenil. Spremenil pa se je način namestitve analitične baze.Za razliko od stare različice, kjer se je analitična baza postavila s pomočjo dokumenta XMLA, kjer so bili podani vsi analitični objekti standardnih modulov BI4Dynamics, se analitična baza nove različice namesti s pomočjo .NET klienta. Vsak objekt analitične baze (dimenzija ali kocka) je v novi arhitekturi zapakiran v svoj dokument XML, ki se v .NET klientu s pomočjo knjižnice AMO, preberejo in namestijo na analitični strežnik. Vsak objekt je strukturiran tako, da se ga lahko na analitični strežnik namesti samostojno.V primeru, da gre za objekt, ki se vgradi v kakšnega od ostalih objektov, je v .NET klientu poskrbljeno za združevanje dokumentov XML znotraj AMO objekta. Združen objekt se nato namesti na analitični strežnik. Na ta način je poskrbljeno za namestitev tudi vseh dodatnih modulov BI4Dynamics.

```
<object BI4DynamicsVersion = "" DataSourceVersion = "" SqlVersion = "" SqlType = "">
    <Dimension xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      <ID>ClosedPeriod</ID>
      <Name>Closed Period</Name>
      <Source xsi:type="DataSourceViewBinding">
        <DataSourceViewID>BI4Dynamics</DataSourceViewID>
      </Source>
      <Attributes>
        <Attribute>
          <ID>ClosedPeriodID</ID>
          <Name>ClosedPeriodID</Name>
          <Usage>Key</Usage>
          <KeyColumns>
            <KeyColumn>
              <DataType>Integer</DataType>
              <Source xsi:type="ColumnBinding">
                <TableID>dim_ClosedPeriod</TableID>
                <ColumnID>ClosedPeriodID</ColumnID>
              </Source>
            </KeyColumn>
          </KeyColumns>
          <AttributeRelationships>
            <AttributeRelationship>
              <AttributeID>Description</AttributeID>
              <Name>ClosedPeriod</Name>
            </AttributeRelationship>
          </AttributeRelationships>
          <OrderBy>Key</OrderBy>
          <AttributeHierarchyVisible>false</AttributeHierarchyVisible>
        </Attribute>
        <Attribute>
          <ID>Description</ID>
          <Name>Closed Period</Name>
          <KeyColumns>
            <KeyColumn>
              <DataType>WChar</DataType>
              <DataSize>100</DataSize>
              <Source xsi:type="ColumnBinding">
                <TableID>dim_ClosedPeriod</TableID>
                <ColumnID>Description</ColumnID>
              </Source>
            </KeyColumn>
          </KeyColumns>
          <OrderBy>Key</OrderBy>
        </Attribute>
      </Attributes>
    </Dimension>
</object>
```

Slika 12: Primer AMO objekta

7 Zaključek

S prenovo arhitekture podatkovnega modela BI4Dynamics smo v podjetju rešili problem pomanjkljive modularnosti stare arhitekture. Pridobitev nove arhitekture se kaže tako na prodajnem kot na razvijalskem področju.

Prodaja je s popolno modularnostjo sistema BI4Dynamics dobila možnost zelo fleksibilnega paketnega načina prodaje. Partnerska mreža BI4Dynamics se razprostira preko 50 držav, kar pomeni, da vsaka od teh držav predstavlja trg s svojimi specifikami. Z združevanjem različnega števila modulov v pakete prodaja lahko zdaj z novo arhitekturo za vsak trg zagotovi paket, ki se bo najlažje prodajal.

S tehničnega vidika nova arhitektura predstavlja zelo dobro ogrodje poslovne inteligence, ki zagotavljaminimalen napor pri razvoju poljubnih analitičnih modulov. Arhitektura je minimalno odvisna od transakcijskega vira podatkov in predstavlja dobro platformo za nadgradnjo rešitve BI4Dynamics tudi na druge transakcijske vire kot sta Microsoft Dynamics AX in Microsoft Dynamics CRM.

Nova arhitektura BI4Dynamics zelo pospeši tudi samo namestitev celotnega analitičnega sistema BI4Dynamics. Pri stari različici sistema BI4Dynamics je namestitev celotnega analitičnega sistema BI4Dynamics trajala tudi dan ali dva. Dolg čas namestitve je bil prisoten predvsem zaradi ročne namestitve dodatnih modulov. Z novo arhitekturo je celoten proces namestitve minimiziran na nekaj minut.

S tehničnega zornega kota je v novi arhitekturi BI4Dynamics še zmeraj nekaj prostora za izboljšave. Tu gre predvsem omeniti nezmožnost vplivanja na strukturo podatkov v samem podatkovnem skladišču BI4Dynamics. Celoten proces izgradnje in polnjenja podatkovnega skladišča je realiziran s skripti SQL, ki pa so zapisani v modulih, kot tekstovni bloki, ki se izvedejo na SQL strežniku. Posledica je, da vsebuje podatkovno skladišče tudi podatke, ki niso relevantni za konkretno implementacijo analitičnega sistema BI4Dynamics. Vsi ti nepotrebni podatki lahko slabo vplivajo na zmogljivostne lastnosti podatkovnega skladišča. Smiselna nadgradnja obstoječe arhitekture bi bil prenos izgradnje logike podatkovnega skladišča kar v .NET klient, ki bi poskrbel za parametrizirano gradnjo vseh skript SQL podatkovnega skladišča.

Nova različica podatkovnega modela je zadovoljila vse trenutne potrebe podjetja. V primeru, da se bodo pojavile dodatne zahteve, pa je zaradi modularne arhitekture odprta za razširitev na dodatne funkcionalnosti.

8 Viri

[1] Data Warehouse. Dostopno na: http://en.wikipedia.org/wiki/Data_warehouse

[2] R. Kimbal, M. Ross, *The Data Warehouse Toolkit, Second Edition*, New York: John Wiley & Sons, 2002, pogl. 1

[3] Introduction to the Unified Dimensional Model (UDM). Dostopno na:http://msdn.microsoft.com/en-us/library/ms345143(SQL.90).aspx

[4] SQL Server Analysis Services (SSAS) – Introduction, Page 1/3. Dostopno na: http://www.sql-server-performance.com/articles/biz/intro_ssas_p1.aspx

[5] SQL Server Analysis Services (SSAS) – Introduction, Page 2/3. Dostopno na: http://www.sql-server-performance.com/articles/biz/intro_ssas_p2.aspx

[6] XML for Analysis. Dostopno na: http://www.xmla.org/

[7] Managed Extensibility Framework. Dostopno na: http://mef.codeplex.com/wikipage?title=Overview&referringTitle=Home

[8] iSlovar. Dostopno na: http://www.islovar.org

[9] AMO Concepts and Object Model. Dostopno na: http://msdn.microsoft.com/en-us/library/bb522603.aspx